Universidad de Cádiz
Escuela Superior de Ingeniería

TRABAJO DE FIN DE MÁSTER

MÁSTER EN SEGURIDAD INFORMÁTICA
(CIBERSEGURIDAD)

SISTEMA DE GESTIÓN DE EVENTOS RELACIONADOS CON ANOMALÍAS EN LA SEGURIDAD EN UN ENTORNO OPEN SOURCE BASADOS EN DOCKER

AUTOR: ALBERTO MEJÍA VITERI

Puerto Real, septiembre 2019

Universidad de Cádiz
Escuela Superior de Ingeniería

TRABAJO DE FIN DE MÁSTER

MÁSTER EN SEGURIDAD INFORMÁTICA
(CIBERSEGURIDAD)

SISTEMA DE GESTIÓN DE EVENTOS RELACIONADOS CON ANOMALÍAS EN LA SEGURIDAD EN UN ENTORNO OPEN SOURCE BASADOS EN DOCKER

DIRECTOR: CARLOS RODRÍGUEZ CORDÓN
AUTOR: ALBERTO MEJÍA VITERI

Puerto Real, septiembre 2019

SISTEMA DE GESTIÓN DE EVENTOS RELACIONADOS CON ANOMALÍAS EN LA SEGURIDAD EN UN ENTORNO OPEN SOURCE BASADOS EN DOCKER

ÍNDICE

Suministrador:
Alberto Mejía Viteri
alberto.mejiaviteri@alum.uca.es

Puerto Real, septiembre 2019

Índice general

Índice de figuras

Índice de tablas

SISTEMA DE GESTIÓN DE EVENTOS RELACIONADOS CON ANOMALÍAS EN LA SEGURIDAD EN UN ENTORNO OPEN SOURCE BASADOS EN DOCKER

MEMORIA

Suministrador:
Alberto Mejía Viteri
alberto.mejiaviteri@alum.uca.es

Puerto Real, septiembre 2019

Memoria

1 Objeto

El objetivo principal de este proyecto es desplegar una solución mediante herramientas open source en un entorno basado en contenedores de Docker para permitir el análisis de anomalías de seguridad. Por lo que se debe recopilar, procesar, y presentar la información relevante mediante visualizaciones en dashboards con el fin de identificar o detectar de manera oportuna incidencias de seguridad en tiempo real.

2 Antecedentes

Los ataques informáticos cada vez son más sofisticados y difíciles de detectar, el gran volumen de información de registros de seguridad generados por los dispositivos red, servidores y aplicaciones no permiten un adecuado análisis por la complejidad de los datos producidos, esto conlleva a no tomar medidas oportunas ante brechas o incidencias de seguridad.

Por otro lado, uno de los principales aspectos para que la gestión de incidentes de las organizaciones tenga éxito es la rapidez con que responden o neutralizan dichos eventos que afectan a la seguridad de la información. El tiempo que transcurre entre la ocurrencia de una incidencia hasta su descubrimiento debe ser lo mínimo posible con el fin de determinar el alcance del incidente y sistemas comprometidos de manera oportuna y eficiente.

Según datos proporcionado por el reporte M-Trends del año 2019 de la empresa FireEye sobre brechas y ciberataques [1], el tiempo promedio global desde que se tiene evidencia de una intrusión hasta la detección es de 78 días, lo que se traduce que un cibercriminal tiene más de dos meses para operar libremente sobre un sistema vulnerable sin que haya sido detectado. Aunque este tiempo se reduce cada año, en dicho reporte señala que el 31% de incidentes son detectados a los 30 días o menos. Este tiempo es todavía alto teniendo en cuenta la capacidad de daño que puede ocasionar a una organización un ataque informático y las pérdidas económicas que puede ocasionar.

Por otra parte, según el reporte elaborado por la Agencia de Seguridad de las Redes y de la Información de la UE (ENISA) [2] el cual se realiza un análisis de los ciberataques, las mayores ciberamenazas para los años 2017 y 2018 siguen siendo Malware, ataques a servicios web, phishing y spam. Esta información se resume en la tabla 1.

Ranking	Ataques 2017	Ataques 2018
1	Malware	Malware
2	Ataques a web	Ataques a web
3	Ataques a aplicaciones	Ataques a aplicaciones
4	Phishing	Phishing
5	Spam	Denegación de servicio (DoS)
6	Denegación de servicio (DoS)	Spam
7	Ransomware	Botnets
8	Botnets	Violación de datos
9	Amenazas internas	Amenazas internas
10	Ataques físicos	Ataques físicos
11	Violación de datos	Fuga de información
12	Robo de identidad	Robo de identidad
13	Fuga de información	Cryptojacking
14	Exploit kits	Ransomware
15	Ciber espionaje	Ciber espionaje

Tabla 1: Ranking de ataques años 2017 y 2018

Un aspecto que llama la atención de este ranking es la identificación de una nueva amenaza presente en el año 2018, el Cryptojacking. Esta ataque se basa en el uso malicioso de ordenadores para la minería de criptomonedas.

Dentro del ámbito de la ciberseguridad, es importante también conocer las principales industrias afectadas por un incidente de seguridad. De acuerdo con el reporte anual realizado por IBM sobre el índice de amenazas [3], las principales organizaciones atacadas en el año 2018 se muestran en la figura 1.

Debido a las diversas amenazas a las que se enfrentan las organizaciones, es fundamental la monitorización y análisis de los registros o eventos generados por los dispositivos, sistemas o aplicaciones, así como de los eventos producidos por propios sistemas operativos relacionados a los registros de auditoría que permiten reconocer inicios de sesión, modificación de ficheros y configuraciones erróneas. Estos mecanismos facilitan la identificación de actividades maliciosas o violaciones en la política de seguridad y permiten tomar medidas de prevención y corrección, esto con el fin de evitar algún incidente de seguridad que afecte a los activos TI.

Por otra parte, muchas de las organizaciones poseen varios métodos simultáneos que permiten la detección de anomalías: sistemas de detección y prevención de intrusos, antivirus, servidores de autenticación, cortafuegos, etc. Esto genera un gran

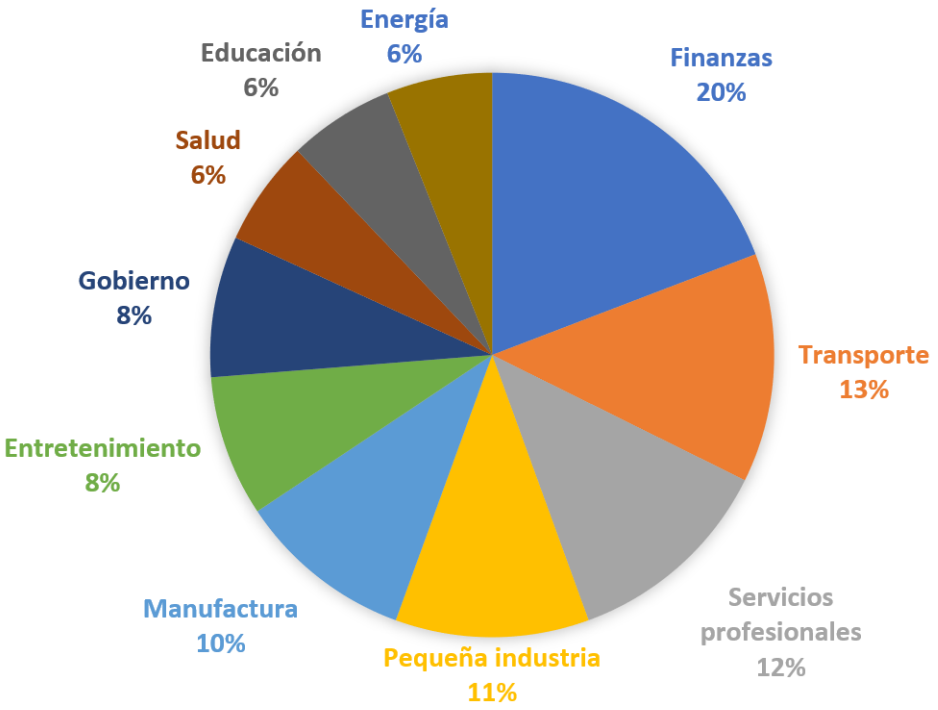


Figura 1: Industrias con mayor cantidad de ataques en año 2018

volumen y variedad de eventos de seguridad, en donde la gestión de esta información se convierte en una tarea complicada de realizar. Muchos de los inconvenientes son relacionados a la capacidad de almacenamiento requerido, diversidad de formatos de los eventos y proporcionar confidencialidad e integridad a estos datos.

Un sistema de gestión eventos relacionados a anomalías trata de solucionar esta tarea de análisis compleja otorgando al analista de seguridad una plataforma que le permita una detección oportuna de anomalías que pueden generar incidencias o brechas de seguridad.

3 Situación actual

Muchas de las organizaciones realizan la recopilación de eventos o logs como elementos indispensables para conocer el comportamiento y estado de la red. La existencia de una gran variedad y cantidad de dispositivos conlleva a que sea necesario el despliegue de un sistema de gestión de logs de forma centralizada, el cual permita al administrador determinar de manera oportuna cualquier tipo de incidencia de algún elemento de la infraestructura de comunicaciones.

Básicamente, existen dos posibilidades en desplegar un sistema de gestión de eventos:

- *Gestión de registros y eventos*: cumple con funciones básicas para la recolección y consolidación de registros de distintas fuentes, normalizar esta información en un formato común, almacenarla y analizarla.
- *Sistema de gestión de información y eventos de seguridad (SIEM):* posee un nivel más completo en el manejo de la información analizada, al proporcionar no sólo la recopilación de eventos, sino establecer un sistema de repuesta de incidentes, generación de alertas e informes.

3.1 Gestión de registros y eventos

Un sistema de gestión de registros consta de varias etapas que permiten el almacenamiento y el análisis de los datos contenidos en los registros. Una de las utilidades en ambientes Linux para la gestión básica de logs de forma centralizada es a través de Rsyslog, el cual posee opciones de filtrado de contenido y enriquecimiento de datos.

Por otro lado, existen otras alternativas donde plataformas se encargan de la gestión integral de eventos que generan una infraestructura de red, de las cuales se destacan las siguientes:

- GrayLog: esta herramienta open source permite la recolección, almacenamiento, enriquecimiento y análisis de logs desde diferentes fuentes. Posee un mecanismo de almacenamiento de registro escalable basado en Elasticsearch, mientras que MongoDB es usado para almacenar metadatos e informacion de configuración. Graylog soporta multiples tipos de entradas: Beats/Logstash, Syslog, GELF, AWS, Netflow, texto plano y CEF. También, es posible integrar con otro tipo de herramientas como por ejemplo Grafana para una visualización gráfica de los datos
- Fluentd: es un colector de logs open source similar a Logstash. Fluentd transforma, analiza y almacena varios tipos de datos. Una de las principales características que tiene esta herramienta es el uso de etiquetas para el manejo de los eventos, estas etiquetas permiten definir donde será el destino de cada evento analizado, a diferencia de Logstash en donde los datos son enviado en un único stream y se elige el destino en base a sentencias condicionales if-then.

  Al igual que Logstash, Fluentd maneja plugins que son accesibles de diversas fuentes y no solamente desde su repositorio oficial, su despliegue puede ser tanto en sistemas Windows como Linux.

- LOGalyze: es un software de código abierto el cual combina la gestión de registros con la monitorización de red con capacidades de recolectar, transformar. almacenar y visualizar datos. Múltiples dispositivos o aplicaciones pueden enviar informacion a esta herramienta mediante el protocolo SOAP. Además, es posible la obtención de reportes y configuración de alertas. Sin embargo, no es una herramienta que proporciona una escalabilidad adecuada y usualmente las sus actualizaciones no soy frecuentes por lo que no es muy popular dentro de las soluciones open source

3.2 Sistema de gestión de información y eventos de seguridad (SIEM)

Existen diversas opciones en el mercado que permiten desplegar un SIEM. Hay varios proveedores dominantes en el mercado de SIEMs, los que destacan:

- IBM
- Splunk
- LogRhythm
- Exabeam
- RSA
- AlienVault
- SolarWinds
- Fortinet
- Logpoint

La compañía Gartner publica un informe anual sobre un análisis de las herramientas de los principales proveedores de estos tipos de soluciones, a través de un cuadrante mágico se evalúan a proveedores en base a dos parámetros:

- *Capacidad de ejecución:* relacionado a factores financieros del proveedor, capacidad de respuesta en el mercado y cartera de clientes
- *Visión integral:* la capacidad de visión del proveedor en tecnología, servicios y funcionalidades de sus soluciones

Los SIEMs en este se clasifican dentro de cuatro grupos:

- *Leaders*: estos proveedores ofrecen soluciones que cumplen con necesidades del mercado de manera satisfactoria y demuestran una capacidad de atender necesidades futuras.

- *Challengers*: proporcionan soluciones adecuadas, son los principales competidores de la grupo anterior, pero no demuestran un cumplimiento de necesidades futuras para impulsar nuevas tecnologías.

- *Visionaries*: estos proveedores poseen soluciones innovadoras, pronostican tendencias en el mercado pero no tiene un cuota de mercado que produzca rentabilidad. Usualmente son adquiridas por proveedores con mayor poder comercial.

- *Niche Players*: soluciones que no posee innovación y se dirige a un segmento particular del mercado. Grandes proveedores pueden estar en este grupo inicialmente cuando con soluciones orientadas a un mercado diferente.

La figura 2 indica el cuadrante mágico que se encuentra el el informe elaborado por Gartner.

Figura 2: Cuadrante mágico de Gartner para SIEMs 2018 [20]

Sin embargo, casi la totalidad de las soluciones mencionadas son comerciales con un elevado costo, por lo herramientas open source son escasas en el mercado. Entre las plataformas open source se tienen los siguientes:

- AlientVault OSSIM: OSSIM es la versión open source de AleintVAult, el cual puede ser una opción adecuada para implementar un SIEM con características básicas en entornos pequeños en donde el flujo de datos no sea significativamente alto.

- Elastic Stack, consta de un conjunto de productos que se pueden integrar de manera sencilla para desplegar un entorno básico de un SIEM, requiere de herramientas adicionales para cumplir con con la funcionalidad completa de un SIEM

- SIEMonster: es una plataforma que integra herramientas open source: RabbitMQ, SearchGuard (seguridad), OSSEC Wazuh (HIDS), OpenAudit y Elastic Stack para recopilar, almacenar y visualizar la información. La versión libre además incluye un sistema de creación de tickets, mientras que el sistema de respuesta a incidentes es implementado solamente en la versión empresarial.

- Prelude: La versión open source es denominada Prelude OSS y posee funcionalidades de un SIEM tal como, colección, normalización, agregación y correlación de eventos. Prelude OSS puede integrarse de forma nativa con herramientas externas:Auditd, LinuxPAM, Suricata, Snort, OSSEC y entre otras. Ademas, posee una APIs y bibliotecas abiertas para integrar fácilmente sensores externos.

  Sin embargo, la versión libre posee un limitado rendimiento lo que lo hace útil solamente para entornos de pruebas o evaluaciones.

4 Normas y referencias

4.1 Disposiciones legales y normativas

UNE 157801:2007 *Criterios generales para la elaboración de proyectos de sistemas de información*

4.2 Bibliografía

[1] *M-TRENDS 2019 Fireeye Mandiante Services Special Report* [en línea]. Disponible en: `https://content.fireeye.com/m-trends`. [Consulta: 21-may-2019]

[2] *ENISA Threat Landscape Report 2018* [en línea].2019. Disponible en: https://www.enisa.europa.eu/publications/enisa-threat-landscape-report-2018. [Consulta: 21-may-2019]

[3] IBM Security. *X-Force Threat Intelligence Index* 2019 [en línea]. Disponible en: https://www.ibm.com/downloads/cas/ZGB3ERYD. [Consulta: 20-may-2019]

[4] Verizon. *2018 Data Breach Investigations Report* [en línea]. 2019. Disponible en: https://www.enisa.europa.eu/publications/enisa-threat-landscape-report-2018. [Consulta: 24-may-2019]

[5] Khalil, George. *Open Source IDS High Performance Shootout* [en línea]. SANS Institute, 2015.

Disponible en: https://www.sans.org/reading-room/whitepapers/intrusion/open-source-ids-high-performance-shootout-35772. [Consulta: 5-jun-2019]

[6] Lukaseder, Thomas; Fiedler, Jessika;Kargl, Frank. *Performance Evaluation in High-Speed Networks by the Example of Intrusion Detection* [en línea]. Disponible en: https://arxiv.org/pdf/1805.11407.pdf. [Consulta: 5-jun-2019]

[7] Saxena, Shilpi. *Practical Real-time Data Processing and Analytics*. Birmingham, 2017.

[8] *Elasticsearch Reference [7.1]* [en línea]. Disponible en: https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html. [Consulta: 6-jun-2019]

[9] *Kibana Reference [7.1]* [en línea]. Disponible en: https://www.elastic.co/guide/en/kibana/current/index.html. [Consulta: 9-jun-2019]

[10] *Logstash Reference [7.1]* [en línea]. Disponible en: https://www.elastic.co/guide/en/logstash/current/index.html. [Consulta: 10-jun-2019]

[11] Berman, Daniel. *Integrating Bro IDS with the ELK Stack - Part 1* [en línea]. 2018. Disponible en: https://logz.io/blog/bro-elk-part-1/. [Consulta: 1-jun-2019]

[12] *Zeek Manual* [en línea]. Disponible en: https://docs.zeek.org/en/stable/index.html. [Consulta: 12-jun-2019]

[13] Smith, Jason; Sanders, Chris . *Applied Network Security Monitoring* [en línea]. Syngress, 2013. Disponible en: https://learning.oreilly.com/library/view/applied-network-security/9780124172081/. [Consulta: 3-jun-2019]

[14] Kumar, Sharath; Shukla, Pranav. *Learning Elastic Stack 6.0* [en línea]. Syngress, 2013. Disponible en: https://learning.oreilly.com/library/view/learning-elastic-stack/9781787281868/. [Consulta: 5-jun-2019]

[15] Bejtlich, Richard. *The Practice of Network Security Monitoring* [en línea]. No Starch Press, 2013. Disponible en: https://learning.oreilly.com/library/view/the-practice-of/9781457185175/. [Consulta: 18-jun-2019]

[16] Bejtlich, Richard. *The Tao of Network Security Monitoring Beyond Intrusion Detection* [en línea]. Addison-Wesley Professional, 2004. Disponible en: https://learning.oreilly.com/library/view/the-tao-of/0321246772/. [Consulta: 27-jun-2019]

[17] Srivastava, Anurag. *Kibana 7 Quick Start Guide* [en línea]. Packt Publishing, 2019. Disponible en: https://learning.oreilly.com/library/view/kibana-7-quick/9781789804034/. [Consulta: 28-jun-2019]

[18] Schenker, Gabriel. *Learn Docker - Fundamentals of Docker 18.x* [en línea]. Packt Publishing, 2018. Disponible en: https://learning.oreilly.com/library/view/learn-docker-/9781788997027/. [Consulta: 28-jun-2019]

[19] Matthias, Karl; Kane, Sean P. *Docker: Up & Running* [en línea]. O'Reilly Media, Inc., 2018. Disponible en: https://learning.oreilly.com/library/view/docker-up/9781492036722/. [Consulta: 1-jul-2019]

[20] Kavanagh, Kelly; Bussa, Toby; Sadowski, Gorka. *Magic Quadrant for Security Information and Event Management* [en línea]. Gartner, 2018. Disponible en: https://virtualizationandstorage.files.wordpress.com/2018/03/magic-quadrant-for-security-information-and-event-3-dec-2018.pdf. [Consulta: 20-jul-2019]

[21] *Comparing AlienVault Unified Security Management to AlienVault OSSIM* [en línea]. 2018. Disponible en: https://cdn5.alienvault.com/docs/whitepapers/comparing-alienvault-usm-to-alienvault-ossim.pdf. [Consulta: 29-jul-2019]

[22] Chuvakin, Anton. *The Complete Guide to Log and Event Management* [en línea]. NetIQ. Disponible en: https://www.microfocus.com/media/white-paper/the_complete_guide_to_log_and_event_management_wp.pdf. [Consulta: 30-jul-2019]

[23] Smulders, Charles. *Magic Quadrants and MarketScopes: How Gartner Evaluates Vendors Within a Market* [en línea]. Gartner, 2011. Disponible en:

https://www.gartner.com/resources/154700/154752/magic_quadrants_and_marketsc_154752.pdf. [Consulta: 01-ago-2019]

[24] *USM Appliance Deployment Guide* [en línea]. AT&T, 2019. Disponible en: https://www.alienvault.com/documentation/resources/pdf/usm-appliance-deployment-guide.pdf. [Consulta: 01-ago-2019]

[25] *Docker Documentation* [en línea]. Disponible en: https://docs.docker.com/. [Consulta: 12-ago-2019]

4.3 Metodología

El despliegue de la solución sera realizados tanto en base a los requerimientos definidos como al estudio de alternativas.Las pruebas realizadas se basarán en la generación de eventos anómalos mediante intentos de intrusiones o explotaciones de vulnerabilidades mediante Kali Linux. La metodología usada se indica en la figura 3

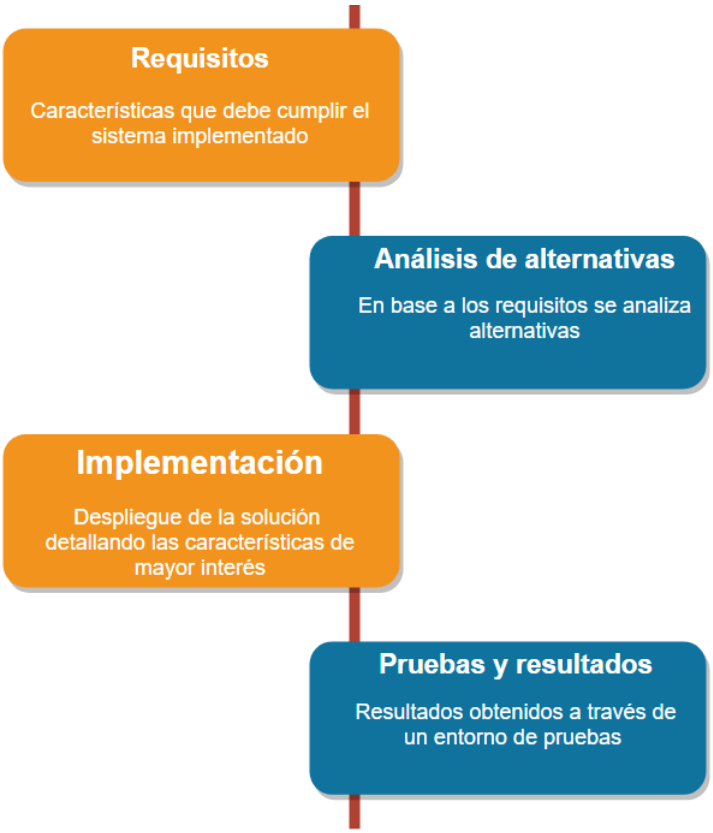


Figura 3: Metodología del proyecto

5 Definiciones y abreviaturas

Definiciones

Agente software que permite realizar una monitorización local y recoge información relevante o métricas de sistemas

Anomalía un evento observable en un sistema o red que se considera fuera de lo común usualmente detectado por un sistema de detección de intrusos o sistema de logs

Exploit método por el cual se realiza un ataque que se aprovecha de las vulnerabilidades de un sistema informático

Hash función que asocia un bloque de datos de entrada a cadenas de bits de longitud fija y única. Los hash pueden ser usados para crear sumas de verificación que validan la integridad de los archivos.

IDS sistema que detecta ataques mediante la captura y análisis de paquetes de red

Incidente de seguridad es la violación de las política de seguridad de un sistema que afectan a la confidencialidad, integridad o disponibilidad de este

Índice Dentro del entorno de Elasticsearch, índice es un espacio de nombres lógico en donde se almacenan datos, tal como ocurre en una base de datos.

Inyección SQL ataque en el que sentencias SQl maliciosas son inyectadas en aplicaciones vulnerables donde no se realiza un adecuada validación de las entradas

IPS sistema con la capacidad de detectar una actividad intrusiva y también detener posibles incidentes

Log un registro de los eventos que ocurren dentro de los sistemas y redes de una organización.

Malware software que está diseñado para dañar u obtener acceso a ordenadores y sistemas informáticos

Métricas medidas utilizadas para analizar y evaluar el rendimiento de un recurso o sistema.

Pipeline término utilizado para describir una serie de ejecuciones secuenciales

Vulnerabilidad debilidad en un sistema de información o en procedimientos de seguridad que podría ser explotado o accionado por una amenaza.

Abreviaturas

API Interfaz de Programación de Aplicaciones

CEF Common Event Format

CSV Comma Separated Values

DNS Domain Name System

DoS Denial of Service

GELF Graylog Extended Log Format

GPL Licencia Pública General

Gbps Gigabits por segundo

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

ICMP Internet Control Message Protocol

IDS Sistema de Detección de Intrusos

IP Internet Protocol

IPS Sistema de Prevención de Intrusos

JSON Notación de Objeto de JavaScript

NIDS Sistema de Detección de Intrusos basados en Red

REST Transferencia de Estado Representacional

SQL Structured Query Language

SSH Secure Shell

SSL Secure Sockets Layer

SOAP Simple Object Access Protocol

TCP Protocolo de Control de Transmisión

TLS Transport Layer Security

UDP Protocolo de Datagrama de Usuario

XML eXtensible Markup Language

6 Requisitos iniciales

A continuación se describe los requisitos que deben cumplir el sistema:

- R-01: Generar anomalías de la red mediante el análisis del tráfico
- R-02: Recopilar datos de contenido completo y de sesión
- R-03: Recopilar datos de estadísticos y de auditoría
- R-04: Enviar eventos desde las fuentes generadoras
- R-05: Normalizar eventos para su almacenamiento
- R-06: Almacenar la información generada
- R-07: Visualizar información para análisis
- R-08: Cifrar datos entre emisor y receptor de eventos
- R-09: Monitorizar el sistema de gestión de eventos

7 Alcance

Forman parte del alcance de este proyecto los siguientes aspectos:

- Análisis de alternativas de herramientas de open source para la gestión de eventos de seguridad.
- Análisis de alternativas de plataformas de análisis de tráfico y detección de anomalías
- Definición de especificaciones del sistema a ser desplegado
- Descripción de la solución propuesta en base a los requerimientos establecidos
- Resultados del sistema implementado

La figura 4 muestra la estructura de descomposición del trabajo del presente proyecto.La finalización de este proyecto dará como resultado de los siguientes entregables:

- Marco teórico
- Memoria
- Anexos
- Especificaciones del sistema
- Presupuesto

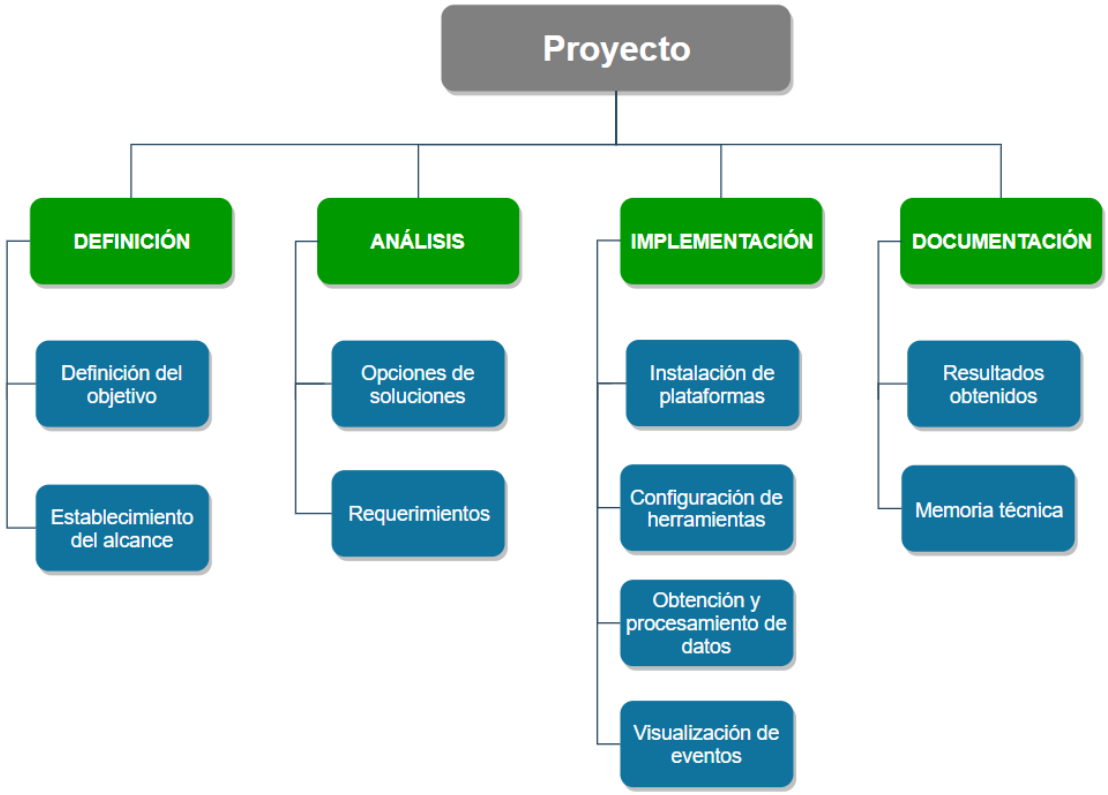


Figura 4: Estructura de descomposición del trabajo

8 Estudio de alternativas y viabilidad

En este apartado se propondrá alternativas que cumplen de manera satisfactoria los requisitos previamente descritos. De manera general, un sistema de gestión de eventos de seguridad debe cumplir las siguientes etapas:

- Recopilación de eventos
- Normalización
- Almacenamiento
- Visualización y análisis

Otro de los aspectos importantes que se debe considerar dentro de un sistema de gestión de eventos es el sistema de detección de anomalías. Este será el encargado de producir los eventos relevantes mediante el análisis del tráfico que circula en la red con el fin de identificar actividades maliciosas.

Las alternativas a compararse serán en base a diferente opciones existentes de un sistema de gestión de información y eventos de seguridad (SIEM) que cumple con las funcionalidades al sistema que se desea desplegar en este proyecto.

Existen diversas alternativas de SIEMs en el mercado, en donde la mayoría de las opciones son plataformas que requieren licenciamiento. Las plataformas más reconocidas son las siguientes:

- QRadar de IBM
- LogRhythm
- Splunk
- Solarwinds LEM
- McAfee Enterprise Security Manager

Además, una de las soluciones lideres de SIEM es AlienteVault. La versión *open source* es denominado AlienVault OSSIM el cual posee algunas de las funcionalidades de las opciones comerciales pero con algunas limitaciones. Por otro lado, Elastic Stack proporciona una adecuada funcionalidad que puede adaptarse a la implementación de un SIEM de una forma rápida y escalable.

8.1 AlienVault OSSIM

Es un SIEM de código abierto que incluye recopilación, normalización, correlación de eventos y detección. OSSIM proporciona un sistema centralizado con el objetivo de monitorizar la actividad de una red e identificar posibles vulnerabilidades, ataques

y comportamiento inusual. Esta información puede ser colacionada para proveer un visión amplia de lo ocurrido en un sistema de información.

OSSIM unifica varias herramientas open source relacionadas a la seguridad y monitorización a través de una interfaz gráfica. Las principales herramientas empleadas en OSSIM son:

- Descubrimientos de activos: Prads y Nmap
- Detección de intrusos: OSSEC, Snort, Suricata y Osiris
- Escaneo de vulnerabilidades: OpenVas, Nikto
- Monitorización: Fprobe, Tcpdump, Nagios
- Datos de sesión: Tcptrack

Las principales funcionalidades que ofrece OSSIM son las siguientes:

- Descubrimiento automático: dispositivos de red son descubiertos a través de sensores para la obtención de información del activo (IP, MAC, sistema operativo, etc). OSSIM solamente permite realizar este descubrimiento de manera local, en el caso de AlientVault USM es posible realizar un descubrimiento en entornos AWS y Microsoft Azure.
- Análisis de vulnerabilidades: se realiza a través de una base de datos de firmas de vulnerabilidades. OSSIM proporciona una base de datos estática, pequeña y sin ningún tipo de actualizaciones por parte de AlienVault, caso contrario ocurre con la opción empresarial en donde se actualiza regularmente la base de datos con nuevas firmas.
- Detección de intrusos: OSSIM proporciona un mecanismo de detección de intrusos mediante reglas de correlación estáticas, básicamente son reglas de ejemplo para que los usuarios implementen las suyas.
- Correlación de eventos: provee un mecanismo de correlación de eventos de seguridad mediante reglas booleanas, esto para proporcionar un mayor enfoque a las posibles actividades maliciosas que ocurre en la red,

8.1.1 Arquitectura

El funcionamiento de OSSIM se basa en tres componentes principales:

- *Sensores*: son empleados para recopilar datos relacionados a la seguridad de diferentes fuentes. Se despliegan en los diferentes segmentos de red para la recolección de logs de activos y monitorizar el tráfico de la red. Los datos son normalizados en un formato común para el posterior envió al servidor. Además, OSSIM soporta la detección de activos de manera automática con el fin de determinar información relacionada al activo escaneado (dirección IP, dirección MAC, sistema operativo, y puertos de escucha).

- *Servidor:* recopila la información proporcionada por los sensores para realizar el procesamiento y correlación de los eventos. Además, realiza tareas de cálculo de riesgos, reconocimiento de patrones y verificación del feed de inteligencia

- *Logger:* almacena la información de los logs en bruto para un posterior análisis forense en el caso de ser necesario.

La figura 5 se indica la arquitectura de la plataforma AlienVault USM.

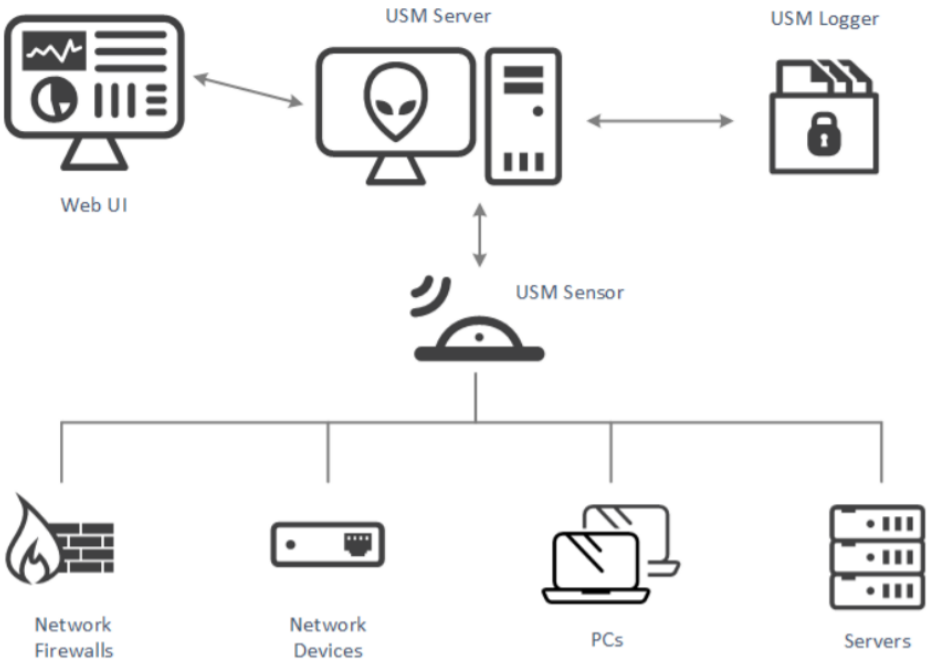


Figura 5: Arquitectura AlienVault [24]

La versión open source (OSSIM) ofrece una menor cantidad de funcionalidades en comparación con la opción empresarial denominada AlienVault USM, por lo que OSSIM es especialmente recomendada para el despliegue dentro de entornos académicos, empresariales pequeños y pruebas de funcionamiento en donde no exista una gran cantidad de flujo de datos.

8.2 Elastic Stack

Elastic Stack consiste en un conjunto de herramientas de código abierto: Elasticsearch, Logstash, Kibana y una familia de agentes (sensores) denominada Beats. Aunque como tal Elastic Stack no es considerada solución que ofrece todas las funcionalidades y componentes que posee un SIEM. Elastic Stack puede ser desplegada de manera conjunta con otras herramientas para convertirlo en una solución SIEM.

La figura 6, indica el esquema de funcionamiento de Elastic Stack, en donde los logs son enviados a Logstash para su normalización, transformación y procesamiento. Elasticsearch indexa y almacena estos datos, una vez realizado esto Kibana permite visualizar, analizar esta información.

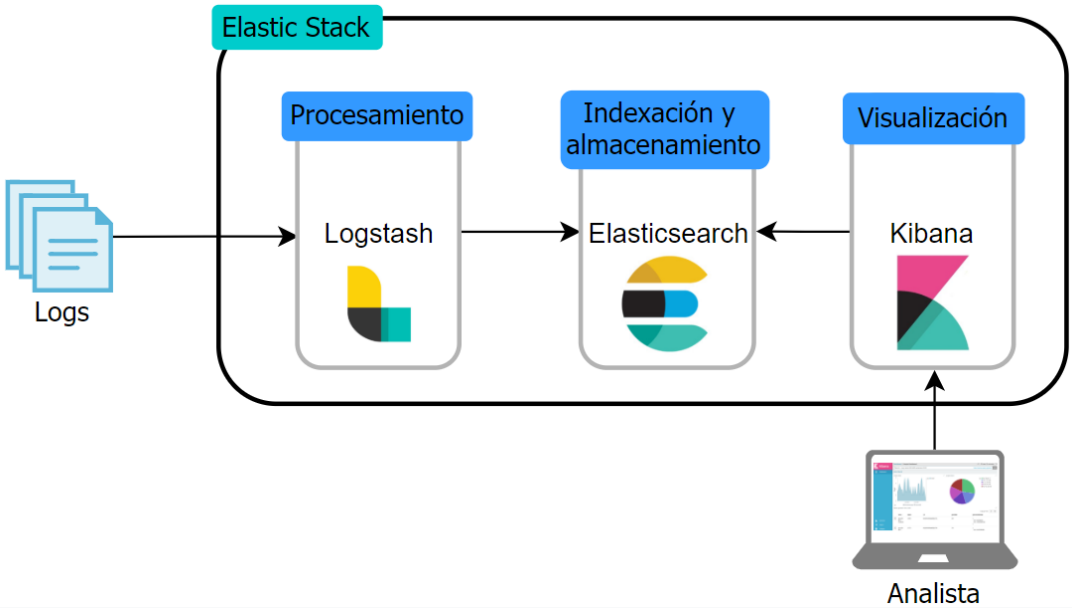


Figura 6: Arquitectura de Elastic Stack

8.2.1 Logstash

Es una herramienta que permite la recolección de logs, análisis sintáctico y almacenamiento de forma centralizada en tiempo real. Logstash utiliza un conjunto de plugins de entrada para recopilar datos de diversas fuentes tales como:

- syslog: mensajes syslog como eventos
- beats: eventos enviados desde agentes de Elastic (filebeat, metricbeat, auditbeat, etc)
- kafka: eventos sobre un tópico en Kafka
- TCP/UDP: eventos sobre conexiones TCP/UDP
- SNMP: sondea dispositivos de red utilizando el protocolo SNMP
- STDIN: eventos desde una entrada estándar

Una vez efectuada la ingesta de estos eventos, Logstash transforma estos datos en un formato común, también existe la posibilidad de añadir información adicional mediante filtros. Finalmente, la información parseada es enviada a una o varias fuentes de almacenamiento tales como Elasticsearch, MongoDB, Hadoop, etc.

Adicionalmente, Logstash permite el manejo de una cantidad aproximadamente de 200 plugins de forma centralizada, en donde estos son accesibles mediante un único repositorio en GitHub. Con respecto la plataforma soportada, esta herramienta permite su despliegue tanto en sistemas Windows como Linux. En el caso de requerir un mecanismo de cifrado de datos que transitan por la herramienta de recopilación, proporciona esta opción para garantizar la seguridad en el trasporte de los datos.

8.2.2 Elasticsearch

Básicamente, Elasticsearch es un motor de búsqueda en un entorno NoSQL, esta basado en la API Apache Lucene y provee una interfaz RESTful compatible con JSON dentro de una arquitectura distribuida. Elasticsearch soporta datos de diferentes fuentes, archivos GitHub, Kafka, Dropbox, Twitter, Wikipedia, etc.

Elasticsearch se enfoca en la mayoría de los casos a realizar analítica de datos, por lo tanto es adecuado para realizar un análisis de eventos acerca de incidentes de seguridad de una manera sencilla y casi en tiempo real. Un analista de seguridad podrá efectuar consultas a los logs en base a los campos de interés tales como direcciones IP, protocolo, longitud del paquetes, etc.

Elasticsearch maneja tanto datos estructurados como no estructurados los cuales son indexados en formato JSON. Además, esta plataforma proporciona una solución fácilmente escalable y una tecnología con alta disponibilidad mediante la replicación de los datos dentro de un cluster y conmutación automática completamente transparente para el usuario.

8.2.3 Kibana

Es una plataforma de visualización de datos para su análisis. Kibana usa una API para consultar o leer información desde los índices de Elasticsearch para realizar búsquedas, visualizar y analizar los datos. Posee diversos tipos de herramientas visualización de los datos, tales como tablas, histogramas, diagramas y mapas. Entre las principales características de esta herramienta estas:

- Exploración y descubrimiento de datos
- Análisis de datos mediante la aplicación de diversas métricas
- Visualización de datos a través de diversos tipos de diagramas
- Monitorización del estado de Elastic Stack
- Generación de reportes
- Creación de alertas en base con valores umbrales
- Cifrado de comunicaciones mediante protocolo TLS

El conjunto de servicios que conforman Elastic Stack hace que sea una opción poderosa para desplegar una solución escalable de gestión de eventos de seguridad, de hecho existen herramientas como es el caso de SIEMoster que se basa en Elasticsearch para la indexación y almacenamiento de datos. Además, recientemente Elastic lanzó su opción para un SIEM, aunque todavía esta en versión beta, es un indicativo de las capacidades que posee Elastic Stack para realizar una analítica de los datos de seguridad en una infraestructura de comunicaciones.

8.3 Análisis de tráfico en tiempo real

Se requiere establecer un mecanismo de análisis de tráfico con el fin de generar eventos relevantes que se produzcan en la red de comunicaciones para su posterior análisis. Como se mencionará en el apartado del estudio teórico, existen diversas herramientas que permiten la inspección de tráfico de red en tiempo real y además

actúan como un sistema de detección o prevención de intrusos (IDS/IPS). Las herramientas *open source* más conocidas son: Snort, Zeek (Bro) y Suricata, por lo que se realizará un breve estudio de estas alternativas.

- **Snort**

  Con respecto a Snort, está diseñada para la detección de tráfico malicioso en la red y puede ser configurada para funcionar en tres modos de operación: *sniffer*, registro de paquetes y sistema de detección de intrusos basados en red (NIDS). Snort es una herramienta simple de implementar por su facilidad de configuración e instalación, ya que su utiliza un lenguaje basado en reglas combinado con un análisis de firmas, este lenguaje es flexible y la creación de nuevas reglas es relativamente sencillo. Aunque esta herramienta ofrece una gran flexibilidad debido a la posibilidad de ser instalada en entornos Windows y Linux, las librerías libpcap y winpcap no permiten el manejo de múltiples hilos. Esto ultimo, provoca que ciertos paquetes no sean correctamente analizados en redes que poseen una alta tasa de transmisión de datos. Aunque Snort cubre esta última deficiencia a través de la versión 3.0, esta es todavía una versión beta.

- **Suricata**

  Suricata al igual que Snort emplea reglas para la detección de tráfico malicioso y es posible la instalación en diversos entornos. La principal diferencia entre estas dos opciones es que Suricata funciona bajo una arquitectura de múltiples hilos, el cual captura y decodifica de manera rápida paquetes de red.Básicamente, la diferencia entre estas dos herramientas podría ser observada en la eficiencia de análisis del tráfico de red, en entornos de grandes volúmenes de datos a alta tasa de transmisión Suricata posee mejor desempeño.

- **Zeek**

  Por otro lado, Zeek además de poder funcionar como un IDS en base a firmas, su motor de análisis genera registros del tráfico analizado basado en un lenguaje scripting. Su principal diferencia con las anteriores herramientas radica en la versatilidad de su lenguaje, el cual a través de su intérprete de políticas permite la personalización de los scripts para indicar que acciones tomar cuando se detecta cierta actividad en la red. Zeek ofrece capacidades útiles en el análisis de protocolos independientemente del puerto, es decir que no requiere conocer el número de puerto para determinar el protocolo usado para una conexión establecida. La configuración y despliegue de esta herramienta

puede ser complejo pero se contrarresta con su capacidad de análisis y buen rendimiento.

Las principales diferencias con respecto a las principales características de cada una de las herramientas se indica en la tabla 2.

Item	Parámetros	Zeek	Snort	Suricata
1	Licencia	BSD	GNU GPL v2	GNU GPL v2
2	Sistema operativo soportado	Unix	Cualquiera	Cualquiera
3	Soporte de tasas de transmisión	Altas	Medias	Altas
4	Hilos	Monohilo	Monohilo	Multihilo
5	Personalización de funcionamiento	Si	No	No
6	Despliegue e instalación	Complejo	Fácil	Fácil
7	Sistema de detección	Basado scripts	Basado en firmas	Basado en firmas

Tabla 2: Comparación de Zeek, Snort y Suricata

Aunque la configuración y despliegue de Zeek puede ser complejo, esta herramienta ofrece grandes ventajas frente al resto. Zeek proporciona una gran capacidad de decodificación e identificación de protocolos en conexiones donde no se establecen sobre los puertos estándar.

Además, Zeek es una plataforma altamente personalizable a través de un lenguaje de script flexible que permite adicionar características propias adaptadas al entorno de despliegue. Estas características resultan interesantes para establecer como sistema de análisis de tráfico a la plataforma Zeek.

9 Descripción de la solución propuesta

9.1 Escenario

La red desplegada sobre un entorno virtual (VirtualBox) se puede observar en la figura 7.

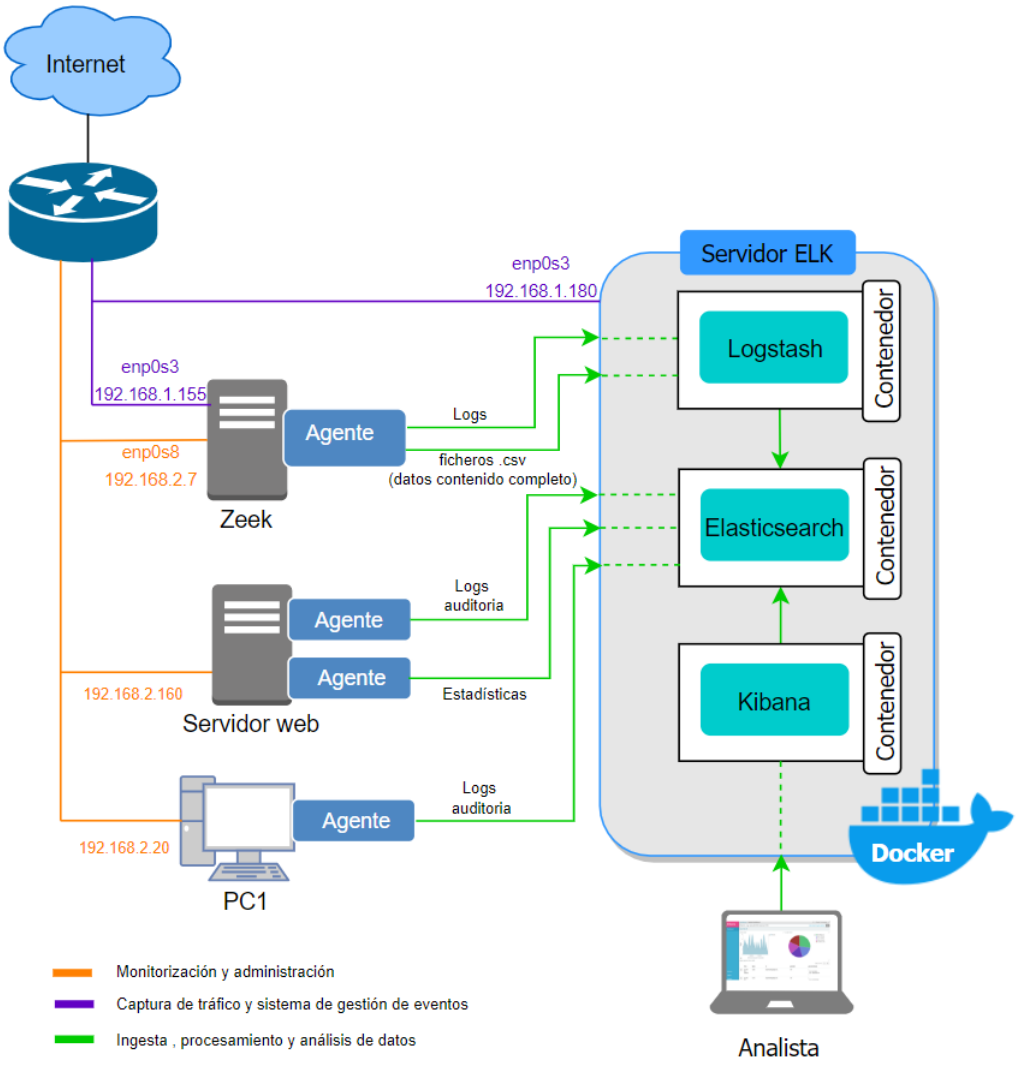


Figura 7: Arquitectura de la solución

Zeek IDS

La herramienta Zeek actúa como el principal generador con el fin de detectar anomalías en la red. Uno de los principales elementos para obtener una copia exacta del tráfico que circula por la red es la utilización de un dispositivo TAP, o a su vez un switch con un puerto espejo que refleje el tráfico circulante hacia Zeek. En este caso, ya que la solución es desplegada en un entorno virtual, será suficiente con la configuración de la tarjeta de red de Zeek en modo promiscuo, esto permite capturar los paquetes que circulan en una determinada subred.

Al momento de establecer un mecanismo de captura de tráfico a través de un IDS,

es importante tomar en cuenta los aspectos de hardware del equipo host para su adecuado funcionamiento.

- CPU: se denomina *worker* al proceso en Zeek que realiza el análisis del tráfico. Todo el procesamiento realizado por los *workers* usan tanto recursos de CPU y memoria. Basado en la documentación de Zeek se define como regla la asignación de 1 núcleo por cada 250 Mbps de tráfico analizado. En el caso de mayor cantidad de trafico combinado es posible conformar un cluster con varios *workers* y núcleos asignados a un mismo equipo físico. La capacidad de análisis de trafico dependerá básicamente de la cantidad de núcleos disponibles en el servidor.

- PCI Bus: definir un adecuado ancho de banda con con el fin de no saturar el bus PCI ante la captura de una gran cantidad de tráfico. Normalmente Virtualbox cuenta con un tipo de bus PCI-X de 64 bits, velocidad de reloj de 133 MHz y un ancho de banda de 1.06 GB/s

- NIC: en ambientes de producción se emplean tarjetas de red con un gran rendimiento y capacidades de filtrado por lo que pueden llegar a ser costosas. En el entorno virtual en el cual se despliega este proyecto no toma en cuenta que exista una tasa alta de captura de datos.

  Por otro lado, se debe evitar el descarte de los datos capturados debido a los mecanismos de segmentación de paquetes (offload) de las NIC . Esto ocasiona que la MTU sea más grande de la permitida por el sensor y que se descarte estos paquetes.

  Puede usar el programa `ethtool`en Linux deshabilitar las características `offload`de una interfaz de red :

```
1       $ ethtool -K nombre_interfaz tso off
```

Las características del servidor son las siguientes:

- Sistema operativo: Ubuntu Server 18.04.2

- Hardware: 4 GB de RAM, 2x CPU y disco 160 GB, NIC Intel PRO/1000 MT (PCI-X)

- Plataformas instaladas: Zeek Bro y Agente de envío de logs

Servidor Elastic Stack

Elasticsearch, Logstash y Kibana se implementan como una plataforma para realizar las funciones de recopilación, normalización, almacenamiento, visualización y análisis de datos en base a los registros o logs generados en la red. Elastic Stack es desplegado sobre un entorno basado en contenedores de Docker. El manejo de contenedores provee varios beneficios, entre los cuales está el uso eficiente de los recursos debido a que no requiere de un sistema operativo para cada carga de trabajo. Además, proporciona soluciones altamente portables y escalables, en donde una solución puede ser desplegada sobre cualquier entono tanto físico, virtual o en la nube y crear réplicas de contenedores de manera fácil.

Las características del servidor Elastic Stack son las siguientes:

- Sistema operativo: Ubuntu Server versión 18.04.2
- Hardware: 2 GB de RAM, 2x CPU y disco 40 GB
- Plataformas instaladas: Docker versión 18.09.5 y Elastic Stack (Logstash, Elasticsearch y Kibana) versión 7.2 en contenedores

Servidor web

Activo a monitorizar mediante la información generada por Zeek y la recolección de varios tipos de datos adicionales a través de agentes instalados localmente.

Este servidor tiene las siguientes características:

- Sistema operativo: Ubuntu Server 18.04.2
- Hardware: 1 GB de RAM, 1x CPU y disco 30 GB
- Plataformas instaladas: Aplicación web MySQL/PHP y Agentes de envío de logs y métricas

PC1

Al igual que el servidor web, es un activo a monitorizar para la detección de anomalías en su funcionamiento mediante la monitorización de Zeek de de manera local. Este servidor tiene las siguientes características:

- Sistema operativo: Ubuntu 18.04.1
- 1 GB de RAM, 1x CPU y disco 20 GB
- Plataformas instaladas: Mozilla Firefox, Agentes de envío de logs

9.1.1 Fuentes de información adicionales

Como se mencionó anteriormente, el sistema de gestión de eventos recopilará información de varias fuentes aparte de Zeek. Estas fuentes adicionales que permitirán complementar la correcta identificación de anomalías de seguridad son:

- **Datos de contenido completo**

  Aunque Zeek realiza la captura del tráfico circulante en la red, no permite obtener la información completa. En este caso se captura los datos de contenido completo a través del *sniffer tshark*, con esta información es posible realizar un análisis con mayor detalle por parte de analista.El *sniffer* será instalado en el servidor Zeek aprovechando la capacidad de escucha del trafico de la red en la interfaz enp0s8.

  Debido a la cantidad de campos que posee estos datos es necesario aplicar filtros para capturar solamente información de interés. Por ejemplo, el siguiente comando permite la captura del tráfico en la interfaz anteriormente mencionada y aplicando filtros para la captura de campos de interés, esta información se transforma a un fichero CSV para la posterior ingesta hacia Logstash.

```
$ tshark -i enp0s8 -lT fields -E header=y -E separator=, -e frame
    .time_epoch -e frame.len -e frame.protocols -e eth.len -e eth.
    src -e eth.dst -e ip.version -e ip.len -e ip.id -e ip.proto -e
     ip.src -e ip.dst -E occurrence=l > packets.csv
```

- **Datos estadísticos**

  Métricas que permiten la supervisión de recursos (CPU, disco, memoria, tráfico de red, etc). La monitorización se realizará de manera local en el servidor web, es decir sin el empleo de protocolos como SNMP o ICMP, sino mediante un agente que recogerá información relevante del servidor y enviarla al sistema de gestión de eventos.

- **Datos de auditoría**

  Permitirá identificar cualquier modificación de ficheros, accesos o inicio de sesión exitosos y fallidos realizados por un determinado usuario, estados de servicios, etc. Los agentes de auditoría serán instalados en el servidor web y en el PC1

Logstash se encargará de la recopilación tanto de logs generados por Zeek como datos de contenido completo para su normalización. Mientras que los datos de auditoría y estadísticos creados en el servidor web y PC1 serán enviados directamente

a Elasticsearch para su indexación ya que no requieren normalización, es decir no es necesario definir filtros para transformar y modificar campos o tipos de datos como ocurre con la información generada por Zeek y datos de contenido completo.

A continuación, se detallada la solución con mayor profundidad para cada uno de los componentes que posee el sistema implementado.

9.2 Análisis de tráfico en tiempo real

Con el fin de detectar algún tipo de ataque en tiempo real se utilizará la herramienta de monitorización de seguridad de red conocida como Zeek basada en la inspección de paquetes. Zeek realiza la búsqueda de ataques conocidos de la misma forma de un IDS. La ventaja de esta herramienta es que las conexiones, sesiones e incluso datos de la capa de aplicación son almacenados en ficheros logs para su posterior procesamiento en un sistema de recopilación de datos.

El funcionamiento por defecto de Zeek es en modo *standalone*, es decir el análisis del tráfico será llevado a cabo por un solo nodo. Dado que Zeek no soporta un manejo mediante múltiples hilos (emplea librería libcap) existe el modo cluster para realizar balanceo de carga y distribuir el tráfico entre varios nodos. En este caso, debido a que no se pretende analizar tráfico a altas tasas de transmisión y volumen, la configuración en modo *standalone* es válida.

Los archivos básicos por configurar en el servidor ubicados en `/usr/local/bro/etc` son los siguientes:

- *node.cfg:* configuración del modo de operación y la interfaz de captura.

```
[zeek]
type=standalone
host=localhost
interface=enp0s8
```

- *networks.cfg:* redes locales que Zeek monitorizará

```
10.0.0.0/8 Private IP space
172.16.0.0/12 Private IP space
192.168.0.0/16 Private IP space
```

- *zeekctl.cfg*: en este fichero se define la configuración global de zeek. También

se puede establecer el tiempo de rotación que tendrá los logs almacenados, en este caso el tiempo de rotación se ha definido en 1 hora (3600 s).

```
# Mail Options
MailTo = alberto.mejiaviteri@alum.uca.es
MailConnectionSummary = 1
MinDiskSpace = 5
MailHostUpDown = 1
# Logging Options
LogRotationInterval = 3600
LogExpireInterval = 0
StatsLogEnable = 1
StatsLogExpireInterval = 0
# Other Options
StatusCmdShowAll = 0
CrashExpireInterval = 0
SitePolicyScripts = local.zeek
LogDir = /usr/local/bro/logs
SpoolDir = /usr/local/bro/spool
CfgDir = /usr/local/bro/etc
```

Tal como se observa en la figura 8 la arquitectura de Zeek consta de tres componentes de los cuales dos son principales: motor de eventos y el intérprete de scripts. Como se mencionó anteriormente, el intérprete de scripts permite definir acciones a llevar a cabo cuando se detecta cierta actividad de acuerdo con las políticas que poseen los scripts.

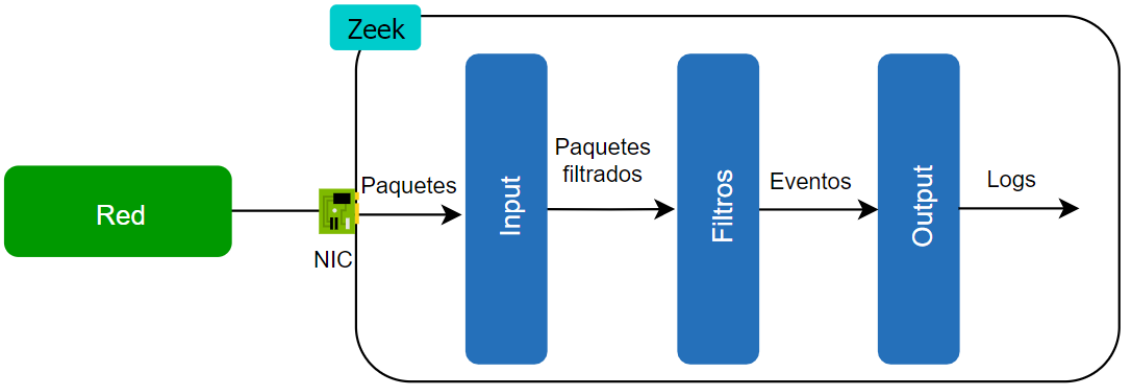


Figura 8: Arquitectura de Zeek

9.2.1 Scripts

Los principales scripts que posee Zeek para la detección de incidencias de seguridad son:

- *scan.bro:* detecta los escaneos de red a un determinado equipo, por ejemplo, utilizado la herramienta nmap. Este script proporciona información importante en el caso de la realización de un reconocimiento activo por parte de un atacante.
- *detect-MHR.bro:* permite comparar los hashes de ficheros reconocidos por Zeek con hashes de malware conocidos con el fin de de identificar una posible amenaza. Este servicio es proporcionado por el grupo Team Cymru y la base de datos de hashes puede ser consultada mediante peticiones Whois (TCP 43), DNS (UDP 53), HTTP (TCP 80) y HTTPS (TCP 443)
- *detect-sqli.bro:* detecta ataques tipo SQL injection
- *detect-bruteforcing.bro:* proporciona información sobre posible ataques mediante fuerza bruta al servicio SSH.

9.2.2 Logs

Zeek almacena información relevante del tráfico analizado en logs en la ubicación `/usr/local/bro/logs/current`, estos logs están escritos en formato ASCCI, organizado en columnas y separado por tabuladores. Los posibles registros o logs generados por Zeek a partir del tráfico analizado son indicados en la tabla 3.

Item	Archivo log	Descripción
1	conn.log	Conexiones TCP, UDC e ICMP
2	dns.log	Peticiones DNS
3	http.log	Solicitudes y respuestas HTTP
4	ssh.log	Conexiones SSH
5	files.log	Análisis de archivos
6	intel.log	Coincidencias de datos del Intelligence Framework
7	notice.log	Notificaciones de detección

Tabla 3: Logs de generados Zeek

Los logs que permitirán la identificación de anomalías a través de Zeek para este proyecto serán los siguientes:

- **intel.log**

  Una de las características más destacables de Zeek es la posibilidad de trabajar con su *Framework* de inteligencia (*Intelligence Framework*), el cual permite consumir datos provenientes de diferentes fuentes con información relevante y actualizada sobre amenazas potenciales. El principal elemento de los datos de inteligencia es el indicador, que puede ser una dirección IP, dirección de correo electrónico, hash de un fichero o certificado, subred, dominio, etc. Además, el indicador contiene un conjunto de metadatos para proporcionar mayor información sobre el evento de seguridad.

  Usualmente, los datos de inteligencia son generados por los procesos de respuesta de incidentes y están disponibles para el uso libre en muchos de los casos. para este proyecto, se producirá datos de inteligencia mediante la integración con un sistema de gestión de amenazas denominado CIF (*Collective Intelligence Framework*), el cual permite combinar información desde diferentes *feeds* (datasets) y consumirlos para poder identificar y detectar actividad maliciosa.

- **notice.log**

  Los eventos de seguridad relacionados a escaneo de puerto, ataques de fuerza bruta, ataques inyección SQL, certificados SSL inválidos, entre otros son enviados a este registro.

- **conn.log**

  Los datos de sesión entre dos nodos son información relavante de seguridad. El registro conn.log almacena todas las conexiones establecidas a través de protocolos de red TCP/UDP e ICMP.

El almacenamiento de los registros pueden ser realizados en un formato JSON facilitando la posterior ingesta de estos logs en la herramienta de recopilación, esto puede ser realizado mediante el uso de un script y cargándolo en el fichero `/local.bro`. Por facilidad, esta modificación de formato será realizada directamente en la herramienta de recopilación de la totalidad de datos de diferentes fuentes.

Además, en el fichero `/local.bro` se define los scripts que debe ejecutar Zeek, a la configuración por defecto se adicionó directivas necesarias para el funcionamiento de datos de inteligencia. A continuación se indica el contenido de este fichero de configuración.

```
@load misc/loaded-scripts
@load tuning/defaults
# Estimar porcentaje de paquetes perdidos
@load misc/capture-loss
# Estadísticas
@load misc/stats
# Escaneos de red
@load misc/scan
# Avisos de software vulnerable
@load frameworks/software/vulnerable
# Detectar cambios de software.
@load frameworks/software/version-changes
# Detecta software usado por varios protocolos
@load protocols/ftp/software
@load protocols/smtp/software
@load protocols/ssh/software
@load protocols/http/software
# Actividad FTP.
@load protocols/ftp/detect
# Monintorizacion de servicios y host.
@load protocols/conn/known-hosts
@load protocols/conn/known-services
@load protocols/ssl/known-certs
# Validación de certificados.
@load protocols/ssl/validate-certs
# Detección de ataques por fuerza bruta SSH.
@load protocols/ssh/detect-bruteforcing
# Detección de SQL injection a.
@load protocols/http/detect-sqli
# Verificación hash de ficheros MD5/SHA.
@load frameworks/files/hash-all-files
# Detección de hash SHA1 en Registro Team Cymru
@load frameworks/files/detect-MHR
## Configuración de datos de inteligencia##
@load frameworks/intel/seen
@load frameworks/intel/do_notice
@load policy/integration/collective-intel
redef Intel::read_files += {
      "/usr/local/bro/ip.intel",
};
```

9.3 Agentes

El objetivo de los agentes es realizar el envío de los datos de las diferentes fuentes que posee el sistema de gestión de eventos de seguridad, por lo tanto es necesario implementar un agente o cliente en las diferentes fuentes que contienen información relevante.

Los agentes permiten el envío de los datos hacia Logstash o Elasticsearch. En este caso, Zeek, el servidor web y PC1 producirán logs o información para ser enviada al sistema de gestión de eventos.

En la figura 9 se ilustra el funcionamiento de los agentes en cada uno de los servidores que producen información relevante y como estos datos son enviados a Logstash y Elasticsearch.

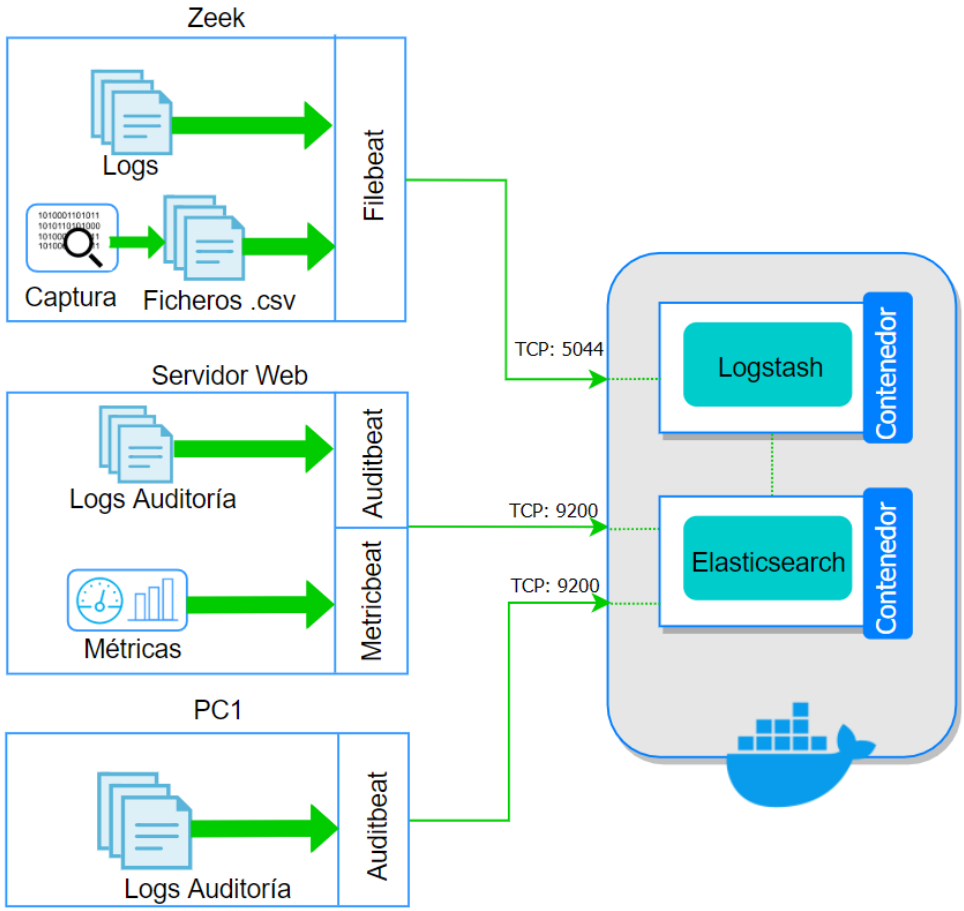


Figura 9: Envío de datos por parte de Agentes

9.3.1 Filebeat

Filebeat es un agente que monitoriza la existencia de registros en una ubicación definida. Recoge los eventos y lo remite a Logstash o Elasticsearch para su procesamiento o indexación. Esta herramienta soporta varios tipos de entradas como log, Syslog, UDP, Netflow, TCP, entre otros.

El principal componente para el funcionamiento de Filebeat es denominado *harvester*, el cual se encarga de abrir y cerrar los ficheros definidos en las entradas. Filebeat crea de manera automática un *harvester* para cada fichero, en donde lee el contenido linea por linea y envía esta información a la salida. Filebeat mantiene el estado del fichero para conocer la última linea del log leída por el *harvester* y asegurar que todas las lineas sean enviadas. Se garantiza que no existan pérdidas de datos en el envío cuando el recurso de salida no este disponible o el servicio haya sido reiniciado.

Filebeat ha sido instalado en el servidor Zeek para la recopilación de información generada por Zeek y datos de contenido completo realizado mediante la captura de tráfico con *tshark*. Es necesario definir los siguientes elementos en la configuración de este agente:

- **Entradas:** se define el tipo de entrada y las rutas en donde se almacenan los registros de interés. La tabla siguiente indica las rutas definidas para los registros generados en Zeek.

Ítem	Log	Ruta	Descripción
1	intel.log	/usr/local/bro/logs/current/intel.log	Información de inteligencia
2	notice.log	/usr/local/bro/logs/current/notice.log	Anomalías detectadas
3	conn.log	/usr/local/bro/logs/current/conn.log	Datos de sesión

Tabla 4: Rutas de registros generados por Zeek

- **Salida:** en la salida de Filebeat será necesario especificar la dirección IP y el puerto de escucha del servidor Logstash, por defecto el puerto 5044.

A continuación se indica el fichero de configuración de Filebeat denominado filebeat.yml, en donde se definen las entradas, salida y opciones de logging.

```
#Entradas
filebeat.inputs:
#LOG1: CONN_ZEEK
- type: log
  enabled: true
  paths:
    - /usr/local/bro/logs/current/conn.log
  fields:
    type: "zeek-conn"
  fields_under_root: true
#LOG2: NOTI__ZEEK
- type: log
  enabled: true
  paths:
    - /usr/local/bro/logs/current/notice.log
  fields:
    type: "zeek-notice"
  fields_under_root: true
#LOG3: INTEL__ZEEK
- type: log
  enabled: true
  paths:
    - /usr/local/bro/logs/current/intel.log
  fields:
    type: "zeek-intel"
  fields_under_root: true

#CSV3:Datos de contenido completo
- type: log
  enabled: true
  paths:
    - /home/packets*.csv
  fields:
    type: "full-packet"
  fields_under_root: true

#Salida Logstash
  hosts: ["192.168.1.180:5044"]
  ssl.certificate_authorities: ["/etc/filebeat/logstash-beat.crt"]

```

```
41 # Logging
42 logging.level: error
43 logging.to_files: true
44 logging.files:
45   path: /var/log/filebeat
46   name: filebeat
47   keepfiles: 7
48   permissions: 0644
```

9.3.2 Auditbeat

La información de auditoría generados por el servidor web serán enviados directamente a Elasticsearch mediante el agente denominado Auditbeat, el cual monitoriza las actividades y procesos de usuarios, así como modificación de ficheros. Auditbeat recopila los datos generados por el *framework* de auditoría de Linux Auditd, esta información se envía a Elasticsearch debido a que no es necesario una normalización, es decir no requiere una procesamiento por parte de Logstash. Auditbeat requiere de la siguiente configuración:

- **Módulos:** los módulos determinan que tipos de métricas son enviadas para sus análisis. Los módulos disponibles para el envío de datos son:
  - Auditd
  - Integridad de ficheros
  - Sistema

  Por ejemplo, para el caso de módulo de integridad de ficheros será necesario definir las rutas de los directorios o archivos que se desean monitorizar. Este módulo por defecto no analiza las rutas definidas en forma recursiva, es decir los cambios realizados subdirectorios no son monitorizados.

- **Salida:** se define la dirección IP del servidor donde se encuentra instalado Elasticsearch y el puerto de escucha (9200).

A continuación se indica el fichero de configuración de Auditbeat denominado auditbeat.yml, en donde se definen las módulos, salida y logging.

```
1 #Modulos
2 auditbeat.modules:
3 - module: file_integrity
4   paths:
5   - /etc/auditbeat
```

```
  - /etc/metricbeat
  - /home/webserver
- module: system
  datasets:
    - host
    - login
    - package
    - process
    - socket
    - user
  # How often datasets send state updates with the
  # current state of the system (e.g. all currently
  # running processes, all open sockets).
  state.period: 12h
  user.detect_password_changes: true

  # File patterns of the login record files.
  login.wtmp_file_pattern: /var/log/wtmp*
  login.btmp_file_pattern: /var/log/btmp*
#Kibana
setup.kibana:
  host: "192.168.1.180:5601"
  username: "xxxx"
  password: "xxxx"
#Outputs
#Elasticsearch output
output.elasticsearch:
  hosts: ["192.168.1.180:9200"]
  username: "xxxx"
  password: "xxxx"
# Logging
logging.level: info
```

9.3.3 Metricbeat

Metricbeat permite realizar una monitorización de manera local en servidores mediante el envío de métricas de rendimiento del sistema, red o servicios específicos. Este agente será utilizado para el envió de información estadística relevante generada por el servidor web.

Los componentes necesario para el despliegue de Metricbeat son:

- **Módulos:** se encarga de definir el servicio a monitorizar, que métricas recopilar, parámetros de conexión y frecuencia de consulta.Etre los principales módulos que soporta Metricbeat se encuentran: Apache, MongoDB, MySQL, Redis, System.

  En este caso se utilizará el módulo *System*, el cual se establece por defecto cuando ningún otro módulo se especifica. Este módulo permite la recolección de métricas relacionados a el uso de CPU, memoria, disco, red y procesos ejecutados.

- **Salidas:** la salida se establece hacia Elasticsearch en el puerto 9200. Al igual que Filebeat, no es necesario la normalización de la información por lo que será enviada directamente a Elasticsearch.

A continuación se indica el fichero de configuración de Metricbeat denominado metricbeat.yml, en donde se definen las modulo, salida y logging.

```
#Modules
metricbeat.config.modules:
  # Glob pattern for configuration loading
  path: ${path.config}/modules.d/*.yml
  # Set to true to enable config reloading
  reload.enabled: false
  # Period on which files under path should be checked for changes
  #reload.period: 10s
# Kibana
setup.kibana:
  username: "xxxx"
  password: "xxxx"
# Outputs
#lasticsearch output
output.elasticsearch:
  hosts: ["192.168.1.180:9200"]
  username: "xxxx"
  password: "xxxx"
#Processors
# Configure processors manipulate events generated by the beat.
processors:
  - add_host_metadata: ~
  - add_cloud_metadata: ~
# Logging
logging.level: info
```

9.4 Contenerización

Tal como se indicó anteriormente serán colocados en contenedores los tres servicios que conforman Elastic Stack:

- Logstash
- Elasticsearch
- Kibana

9.4.1 Imagen Docker

Los contenedores se basan en una imagen, la cual se podría considerar el sistema de archivos del contenedor. Una imagen contiene librerías, variables de entorno y ficheros de configuración necesarios para ejecutar una aplicación, en este caso Elastic Stack.

La imagen utilizada para este proyecto se encuentra en el repositorio de imágenes denominada Docker Hub en la dirección `https://hub.docker.com/r/sebp/elk/`, esta imagen ha sido modificada para adaptarse al entorno requerido.

9.4.2 Volúmenes

Con el fin de mantener los archivos creados dentro del contenedor cuando este sea eliminado es necesario la creación de volúmenes. Los volúmenes se almacenan en alguna parte del sistema de archivos de la máquina host donde se ejecuta Docker. En este caso, cada volumen creado debe corresponder a la ruta de directorios o ficheros de configuración necesario para el funcionamiento de los servicios que conforman Elastic Stack en cada contenedor.

A la imagen original fue agregado un volumen adicional a cada servicio, el cual contiene certificados para establecer una comunicación segura entre estos. A continuación, se describe los volúmenes creados para cada contenedor en la tabla 5.

En la figura 10 se puede observar de una mejor manera como se han creado los volúmenes en Docker para cada uno de los servicios que conforman Elastic Stack.

Fuente (Host)	Destino (Docker)
./logstash/config/logstash.yml	/usr/share/logstash/config/logstash.yml
./logstash/pipeline	/usr/share/logstash/pipeline
./logstash/config/certs	/usr/share/logstash/config/certs
./elasticsearch/config/elasticsearch.yml	/usr/share/elasticsearch/config/elasticsearch.yml
./elasticsearch/config/certs	/usr/share/elasticsearch/config/certs
./kibana/config/kibana.yml	/usr/share/kibana/config/kibana.yml
./kibana/config/certs	/usr/share/kibana/config/certs

Tabla 5: Volúmenes en Docker

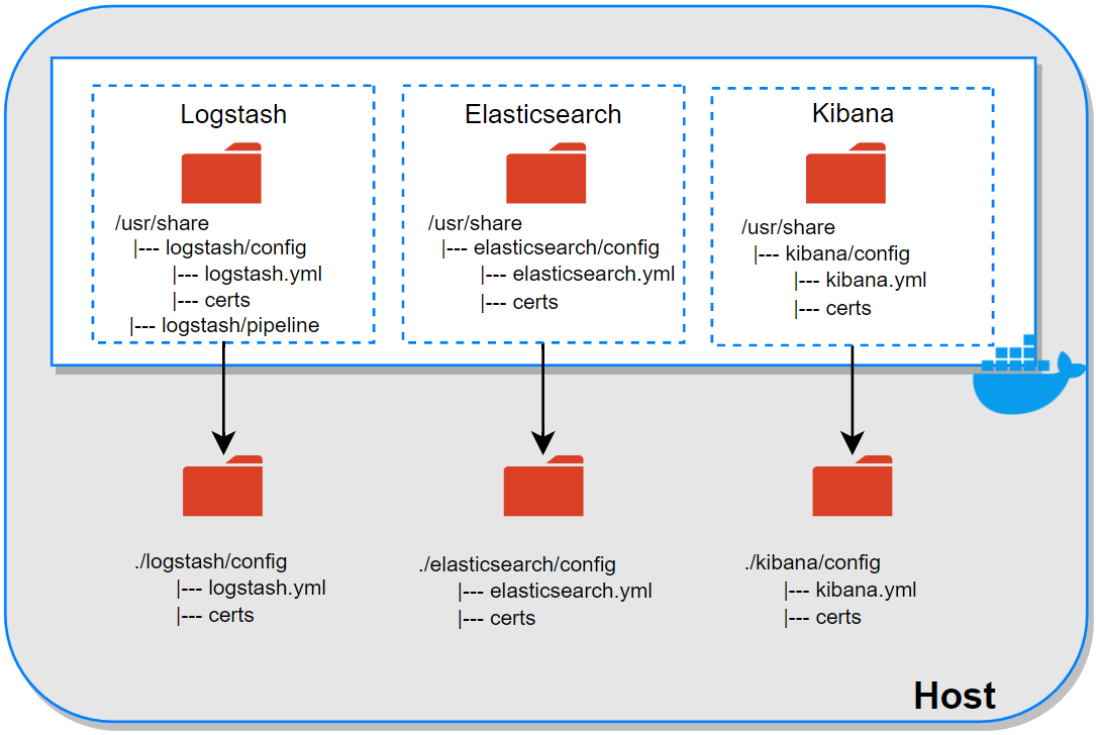


Figura 10: Volúmenes creados en Docker

Las configuraciones establecidas a cada contenedor y volúmenes son definidos en un fichero de denominado *docker-compose.yml*. En este fichero también se define el mapeo de puertos, versión de cada servicio y configuración de red. La siguiente

salida muestra el contenido de este fichero.

```
version: '2'
services:
# Elasticsearch
  elasticsearch:
    build:
      context: elasticsearch/
      args:
        ELK_VERSION: $ELK_VERSION
    volumes:
      - ./elasticsearch/config/elasticsearch.yml:/usr/share/elasticsearch/config/elasticsearch.yml:ro
      - ./elasticsearch/config/certs:/usr/share/elasticsearch/config/certs:ro
    ports:
      - "9200:9200"
      - "9300:9300"
    environment:
      ES_JAVA_OPTS: "-Xmx256m -Xms256m"
      ELASTIC_PASSWORD: xxxx
    networks:
      - elk
# Logstash
  logstash:
    build:
      context: logstash/
      args:
        ELK_VERSION: $ELK_VERSION
    volumes:
      - ./logstash/config/logstash.yml:/usr/share/logstash/config/logstash.yml:ro
      - ./logstash/pipeline:/usr/share/logstash/pipeline:ro
      - ./logstash/certs:/usr/share/logstash/certs:ro
    ports:
      - "5044:5044"
      - "5045:5045"
      - "9600:9600"
    environment:
      LS_JAVA_OPTS: "-Xmx256m -Xms256m"
    networks:
      - elk
```

```
    depends_on:
      - elasticsearch
# Kibana
  kibana:
    build:
      context: kibana/
      args:
        ELK_VERSION: $ELK_VERSION
    volumes:
      - ./kibana/config/kibana.yml:/usr/share/kibana/config/kibana.yml:ro
      - ./kibana/config/certs:/usr/share/kibana/config/certs:ro
    ports:
      - "5601:5601"
    networks:
      - elk
    depends_on:
      - elasticsearch

networks:

  elk:
    driver: bridge
```

La versión instalada de los servicios que conforman Elastic Stack será la 7.2.1 que corresponde a la última versión estable de la plataforma. Mientras que el mapeo de puertos está relacionado a la publicación de puertos para poner a disposición de los servicios fuera del contenedor. Por ejemplo, para establecer la conexión desde los agentes. Los puertos TCP publicados son los siguientes:

- 9200: Elasticsearch
- 5601: Kibana
- 5044: Logstash

La creación y ejecución de los contenedores a partir del archivo docker-compose.yml es realizado con la herramienta Docker Compose, mediante el comando *docker-compose up*. El comando *docker ps* permite comprobar los contenedores que se están ejecutando en la maquina host, el resultado se indica en la siguiente salida.

```
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
fd44bd422991 dockerelk_logstash "/usr/local/bin/…dock" 11 hours ago Up 8 hours 0.0.0.0:5044-5045->5044-5045/tcp, 0.0.0.0:9600->9600/tcp dockerelk_logstash_1
9d785757ab80 dockerelk_kibana "/usr/local/bin/…kiba" 12 hours ago Up 9 hours 0.0.0.0:5601->5601/tcp dockerelk_kibana_1
103f4ffe916f dockerelk_elasticsearch "/usr/local/bin/…dock" 12 hours ago Up 9 hours 0.0.0.0:9200->9200/tcp, 0.0.0.0:9300->9300/tcp dockerelk_elasticsearch_1
```

9.4.3 Recopilación de información

Dentro del contenedor, Logstash actuará como el motor de recopilación de los logs enviados por el agente Filebeat. La principal finalidad de Logstash es proporcionar un mecanismo de normalización de los datos generados por Zeek y de contenido completo mediante la modificación del formato.

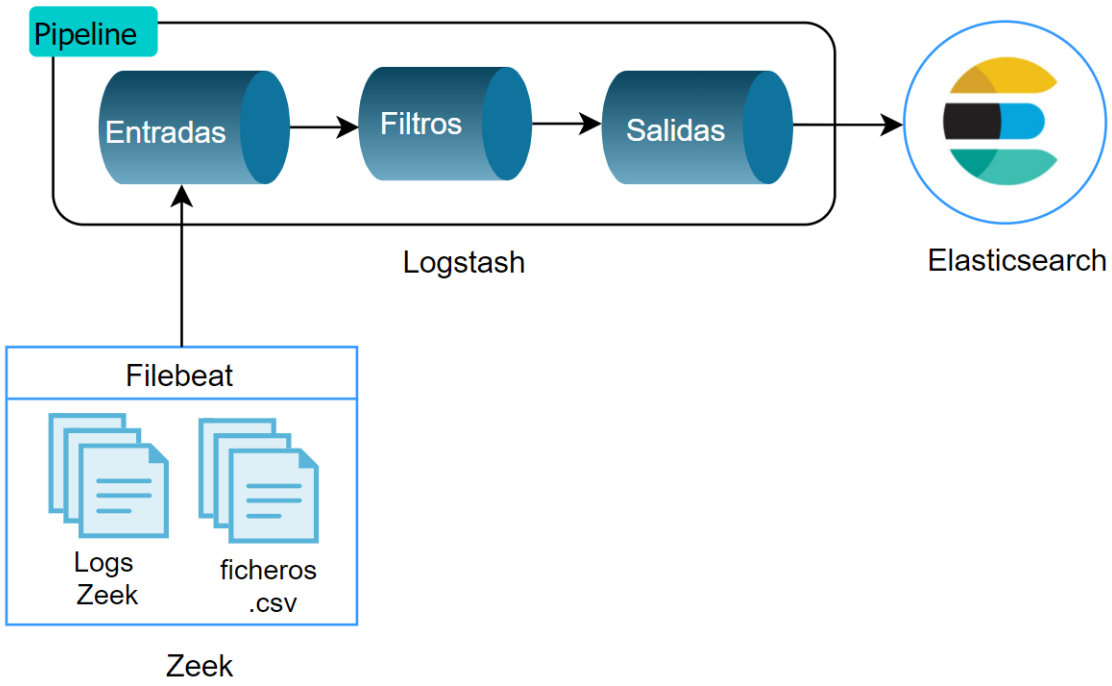


Figura 11: Arquitectura Logstash

Como se observa en la figura 11 el procesamiento de la información recibida por Logstash se basa en la creación de un pipeline, el cual está conformado por:

- Entradas
- Filtros
- Salidas

Cada uno de los bloques que conforman el pipeline deben ser definidos en un archivo de configuración dentro del directorio de Logstash.

Entradas

Permiten establecer la fuente de los logs enviados, en este caso es necesario definir que esta información es enviada desde el agente Filebeat a través del puerto TCP de escucha 5044, puerto publicado en el contenedor para el establecimiento de la conexión entre Filebeat y Logstash.

Filtros

Los filtros realizan el procesamiento de los eventos con la finalidad de transformar los datos en un formato definido o normalizarlos. Básicamente la complejidad en la implementación del pipeline radica en el bloque de filtros. Por ejemplo, los siguientes pasos son realizados en esta etapa para el caso de los registros conn.log:

1. Eliminación de comentarios ubicados en la parte superior de los ficheros de logs. Se utiliza expresiones regulares para eliminar todas las líneas que empieza con el caracter numeral (#).

2. Eliminación de los tabuladores mediante el filtro CSV para realizar la trasformación de los datos. Aunque la separación de las columnas no sea realizada por comas, CSV permite el manejo de tabuladores dentro de un fichero.

3. Transformación de la marca de tiempo (*timestamp*) que establece Zeek de tipo UNIX a tipo fecha, esto se realiza empleado un filtro que posee Logstash. Un ejemplo de la aplicación de este filtros es el siguiente:

   1560212576.046810 -> 11 Jun, 2019 0:22:56.046

4. Finalmente, es necesario realizar dos tareas: modificar los nombres de los campos y los tipos de datos.

   - Es necesario el empleo del filtro denominado *Mutate* para modificar los nombres de campos que contienen puntos, debido que Logstash presenta ciertos inconvenientes en procesar estos caracteres. Por ejemplo, se realizará el siguiente cambio para los nombres de los campos para los logs `conn.log`.

  - id.orig_h => id_orig_host
  - id.orig_p => id_orig_port
  - id.resp_h => id_resp_host
  - id.resp_p => id_resp_port
- Al mismo tiempo, Zeek soporta en su lenguaje scripting tipos de datos: enum, port, addr, etc. Por lo que debe modificar estos tipos a *float* e *integer* en los campos necesarios. Por ejemplo:
  - id.orig_p => integer
  - id.resp_p => integer
  - orig_bytes => integer
  - duration => float
  - resp_bytes => integer
  - missed_bytes => integer

A continuación se indica los filtros creados para cada uno de los ficheros que Logstash procesa para la normalización del formato de los datos.

```
#Eliminación de comentarios
filter {
  if [message] =~ /^#/ {
    drop { }
  }
#Transformaciones para conn.log
## Establecer nombres de campos
  if [type] == "zeek-conn" {
    csv {
      columns => ["ts","uid","id.orig_h","id.orig_p","id.resp_h","id.resp_p","proto","service","duration","orig_bytes","resp_bytes","conn_state","local_orig","local_resp","missed_bytes","history","orig_pkts","orig_ip_bytes","resp_pkts","resp_ip_bytes","tunnel_parents"]

      separator => " "
    }
## Cambio de timestamp a tipo fecha
    date {
      match => [ "ts", "UNIX" ]
    }
```

```
# Modificación de nombres de campos y tipos de datos
    mutate {
      convert => { "id.orig_p" => "integer" }
      convert => { "id.resp_p" => "integer" }
      convert => { "orig_bytes" => "integer" }
      convert => { "duration" => "float" }
      convert => { "resp_bytes" => "integer" }
      convert => { "missed_bytes" => "integer" }
      convert => { "orig_pkts" => "integer" }
      convert => { "orig_ip_bytes" => "integer" }
      convert => { "resp_pkts" => "integer" }
      convert => { "resp_ip_bytes" => "integer" }
      rename => { "id.orig_h" => "id_orig_host" }
      rename => { "id.orig_p" => "id_orig_port" }
      rename => { "id.resp_h" => "id_resp_host" }
      rename => { "id.resp_p" => "id_resp_port" }
    }
    fingerprint{
      method => "SHA1"
      key => "xxxx"
    }
  }
#Transformaciones para notice.log
## Establecer nombres de campos
  if [type] == "zeek-notice" {
    csv {
      columns => ["ts","uid","id.orig_h","id.orig_p","id.resp_h","id.↩
          ↪ resp_p","fuid","file_mime_type","file_desc","proto","note",↩
          ↪ "msg","sub","src","dst","p","n","peer_descr","actions","↩
          ↪ suppress_for","dropped","remote_location.country_code","↩
          ↪ remote_location.region","remote_location.city","↩
          ↪ remote_location.latitude","remote_location.longitude"]
      separator => " "
    }
## Cambio de timestamp a tipo fecha
    date {
      match => [ "ts", "UNIX" ]
    }
# Modificación de nombres de campos y tipos de datos
    mutate {
      convert => [ "id.orig_p", "integer" ]
```

```
54      convert => [ "id.resp_p", "integer" ]
55      convert => [ "p", "integer" ]
56      convert => [ "n", "integer" ]
57      convert => [ "suppress_for", "float" ]
58      rename => [ "id.orig_h", "id_orig_host" ]
59      rename => [ "id.orig_p", "id_orig_port" ]
60      rename => [ "id.resp_h", "id_resp_host" ]
61      rename => [ "id.resp_p", "id_resp_port" ]
62      rename => [ "remote_location.country_code", "↩
            ↪ remote_location_country_code" ]
63      rename => [ "remote_location.region", "remote_location_region" ]
64      rename => [ "remote_location.city", "remote_location_city" ]
65      rename => [ "remote_location.latitude", "remote_location_latitude"↩
            ↪  ]
66      rename => [ "remote_location.longitude", "↩
            ↪ remote_location_longitude" ]
67    }
68    fingerprint{
69      method => "SHA1"
70      key => "xxxx"
71    }
72  }
73
74#Transformaciones para intel.log
75## Establecer nombres de campos
76 if [type] == "zeek-intel" {
77    csv {
78      columns => ["ts","uid","id.orig_h","id.orig_p","id.resp_h","id.↩
            ↪ resp_p","seen.indicator","seen.indicator_type","seen.where"↩
            ↪ ,"seen.node","matched","sources","fuid","file_mime_type","↩
            ↪ file_desc"]
79      separator => " "
80    }
81## Cambio de timestamp a tipo fecha
82    date {
83      match => [ "ts", "UNIX" ]
84    }
85# Modificación de nombres de campos y tipos de datos
86    mutate {
87      convert => [ "id.orig_p", "integer" ]
88      convert => [ "id.resp_p", "integer" ]
```

```
89      convert => [ "matched", "float" ]
90      rename => [ "id.orig_h", "id_orig_host" ]
91      rename => [ "id.orig_p", "id_orig_port" ]
92      rename => [ "id.resp_h", "id_resp_host" ]
93      rename => [ "id.resp_p", "id_resp_port" ]
94      rename => [ "seen.indicator", "seen_indicator" ]
95      rename => [ "seen.where", "seen_where" ]
96      rename => [ "seen.node", "seen_node" ]
97    }
98    fingerprint{
99      method => "SHA1"
100     key => "xxxx"
101   }
102 }
103#Transformaciones para datos de contenido completo (fichero .csv)
104## Establecer nombres de campos
105 if [type] == "full-packet" {
106   csv {
107     columns => ["frame.time_epoch","frame.len","frame.protocols","eth.↩
          ↪ len","eth.src","eth.dst","ip.version","ip.len","ip.id","ip.↩
          ↪ proto","ip.src","ip.dst"]
108     separator => ","
109   }
110## Cambio de timestamp a tipo fecha
111   date {
112     match => [ "frame.time_epoch", "UNIX" ]
113   }
114# Modificación de nombres de campos y tipos de datos
115   mutate {
116     rename => [ "frame.time_epoch", "frame_time_epoch" ]
117     rename => [ "frame.len", "frame_len" ]
118     rename => [ "frame.protocols", "frame_protocols" ]
119     rename => [ "eth.len", "eth_len" ]
120     rename => [ "eth.src", "eth_src" ]
121     rename => [ "eth.dst", "eth_dst" ]
122     rename => [ "ip.version", "ip_version" ]
123     rename => [ "ip.len", "ip_len" ]
124     rename => [ "ip.id", "ip_id" ]
125     rename => [ "ip.proto", "ip_proto" ]
126     rename => [ "ip.src", "ip_src" ]
127     rename => [ "ip.dst", "op_dst" ]
```

```
    }
    fingerprint{
      method => "SHA1"
      key => "xxxx"
    }
  }
}
```

Salida

En esta etapa se define el destino de la información transformada. Tan pronto como Logstash realiza la transformación de los eventos mediante los filtros, estos datos son enviados a Elasticsearch para su almacenamiento e indexación. En este bloque de salida se requiere definir un nombre de índice (ver tabla 6, el cual nos permitirá agregar y visualizar los datos en Kibana. El nombre de índice tendrá el formato `filebeat-yyyy.MM.dd`, ya que los datos recibidos de Logstash son enviados por Filebeat.

Conexión Filebeat-Logstash

Existe la posibilidad de verificar la configuración de la salida de Filebeat y la comprobación de la conexión hacia Logstash a través del uso del comando

```
$ sudo filebeat test output
```

Un resultado exitoso será el siguiente:

```
logstash: logstash:5044...
  connection...
    parse host... OK
    dns lookup... OK
    addresses: 192.168.1.180
    dial up... OK
  TLS... WARN secure connection disabled
  talk to server... OK
```

Como se puede observar el resultado del comando indica que la comunicación entre Filebeat y Logstash no posee una autenticación basada en TLS/SSL que permita una conexión segura entre estas dos entidades. Uno de los aspectos a considerar con el fin de garantizar una mayor seguridad en la red es proporcionar comunicaciones cifradas entre los servicios que conforman el sistema de gestión de logs.

Es posible proteger la comunicación a través del protocolo SSL, proporcionando de esta manera confidencialidad a las comunicaciones. Por lo tanto, será necesario la creación de un certificado auto firmado mediante una clave privada en el servidor. En este caso Logstash actuará como servidor y Filebeat como cliente dentro entorno SSL/TLS. Se empleará el estándar X.509 para la generación del certificado y llave privada a través del paquete de herramientas OpenSSL. Los ficheros generados tendrán una extensión .key y .crt. Estos ficheros serán almacenados en el servidor (Logstash) y solamente el fichero con extensión .crt, que representa la parte pública será distribuido y almacenado en el cliente (Filebeat).

Antes de arrancar el servicio Filebeat es necesario validar el certificado en el servidor mediante el comando:

```
curl -v --cacert ca.crt http://192.168.1.180:5044
```

Un resultado exitoso de la validación es similar a:

```
...
GET / HTTP/1.1
> Host: 192.168.1.180:5044
> User-Agent: curl/7.58.0
> Accept: */*
>
* TLSv1.2 (IN), TLS alert, Client hello (1):
* Empty reply from server
* Connection #0 to host 192.168.1.180 left intact
curl: (52) Empty reply from server
```

Posteriormente al inicio del servicio Filebeat, se puede verificar la conexión y salida de Filebeat hacia Logstash mediante:

```
$ sudo filebeat test output
```

El resultado del comando anterior y donde se puede visualizar el establecimiento de una comunicación cifrada entre los dos servicios es el siguiente:

```
logstash: 192.168.1.180:5044...
  connection...
    parse host... OK
    dns lookup... OK
    addresses: 192.168.1.180
    dial up... OK
```

```
TLS...
  security: server's certificate chain verification is enabled
  handshake... OK
  TLS version: TLSv1.2
  dial up... OK
talk to server... OK
```

9.4.4 Almacenamiento e indexación

Elasticsearch proporciona la capacidad de almacenamiento e indexación de la información enviada por Logstash, trabaja esencialmente sobre solicitudes tipo HTTP y datos tipo JSON contra su API REST. Los datos son almacenados en uno o varios índices de manera análoga con una base de datos.

Esta herramienta requiere de una reducida configuración, solamente es necesario definir la dirección IP por la cual los servicios pueden comunicarse con Elasticsearch, así como su número de puerto. Por defecto Elasticsearch usa una dirección *loopback* y numero de puerto 9200. Se debe modificar la configuración de red en el archivo de configuración de Elascticsearch para que sea alcanzable por los agentes Filebeat, Metricbeat y Auditbeat.

Aunque como se detalló anteriormente Filebeat no envía ninguna información de logs directamente a Elasticsearch, pero es indispensable que Filebeat cargue la plantilla de índices con el fin de indicar la forma de procesar los campos que contienen los logs. La carga de la plantilla de índices se debe realizar manualmente debido a que Filebeat lo hace de forma automática solamente cuando este envía directamente información a Elasticsearch, recordando lo observado en la figura 11 el plugin Filebeat envía información a Logstash no a Elasticsearch. El comando empleado para realizar la carga de la plantilla desde Filebeat hacia Elasticsearch es:

```
$ filebeat setup --template -E output.logstash.enabled=false -E 'output.
    elasticsearch.hosts=["192.168.1.180:9200"]'
```

Siendo la IP 192.168.1.180 la dirección donde está instalado Elasticsearch, el puerto de escucha 9200 y considerando que exista conectividad entres estas dos herramientas a través de la configuración en los campos relacionados con la red realizados en Elasticsearch.

Como se explicó anteriormente, los datos recogidos por Metricbeat y Auditbeat son enviados directamente a Elasticsearch sin ser proesados por Logstash, esto debido

que esta información no requiere ser normalizada ni realizar ninguna modificación de tipos de datos y nombres de campos como ocurre con logs generados por Zeek y ficheros .csv con datos de contenido completo.Los agentes almacenan los eventos en los índices bajo el nombre y con el formato que se indican el la tabla siguiente.

Agente	Formato del nombre de índice
Filebeat	filebeat-yyyy.MM.dd
Metricbeat	metricbeat-7.2.0-yyyy.MM.dd
Auditbeat	auditbeat-7.2.0-yyyy.MM.dd

Tabla 6: Nombres de índices de Elasticsearch

Con el fin de validar si la información es correctamente enviada a Elasticsearch desde Filebeat, se puede ejecutar la siguiente petición GET:

```
$ curl -X GET http://localhost:9200/filebeat*/_search?pretty
```

Y se tendrá la siguiente resultado:

```
{
  "took" : 8,
  "timed_out" : false,
  "_shards" : {
    "total" : 8,
    "successful" : 8,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 6277,
      "relation" : "eq"
    },
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "filebeat-2019.06.08",
        "_type" : "_doc",
        "_id" : "eSlPOWsB1RCn10CFKDjO",
        "_score" : 1.0,
        "_source" : {
```

```
        "agent" : {
          "ephemeral_id" : "28aa5dba-779f-490f-91c4-303c1e4d5c19",
          "hostname" : "zeek",
          "id" : "91d7c671-6681-4c3c-a3f0-5c44a5e3e9d2",
          "version" : "7.1.1",
          "type" : "filebeat"
        },
        "@timestamp" : "2019-06-08T22:38:29.005Z",
        "type" : "zeek-conn",
        "message" : "1560033498.011134\tCusRPZO2LqTJ4i6gb\t192↩
            ↪ .168.1.150\t58943\t8.8.8.8\t53\tudp\tdns\t0.350256\t0\↩
            ↪ t125\tSHR\tT\tF\t0\tCd\t0\t0\t1\t153\t-",
        "log" : {
          "offset" : 2727,
          "file" : {
            "path" : "/usr/local/bro/logs/current/conn.log"
          }
          ...
```

La salida anterior indica que el índice contiene objetos JSON relacionados a los campos de los logs. Además, se puede apreciar que el contenido de los logs está en el campo "message". Para listar los índices disponibles en Elasticsearch se puede emplear ejecutar la siguiente petición:

```
curl -X GET http://localhost:9200/_cat/indices?v
```

En el resultado se puede observar los índices creados a partir de los eventos enviados por los Agentes

```
health status index                            uuid                   pri rep docs.count
yellow open   metricbeat-7.2.0-2019.07.05 aTfDWxOTSr-DA6nXGxb_DA   1   1      21754
green  open   .monitoring-es-7-2019.07.05 JufX9IMERLOR8NwnB7QN4g   1   0      67993
yellow open   filebeat-2019.07.11          fj4qOfRaTZ6UyehyIxzJpw   1   1        610
yellow open   auditbeat-7.2.0-2019.07.04   LaO870q5QRyRH-hIKS8ZWQ   1   1       6080
yellow open   filebeat-2019.07.07          2SqvUimZTo2RBHrss9dbvw   1   1          6
green  open   .security-7                  dpRbBxaISS-rw2IG4lZI8A   1   0          4
yellow open   filebeat-2019.07.09          2_L_8XprRDytEC2n39Jrxw   1   1      19081
```

Figura 12: Índices en Elasticsearch

9.5 Exploración y visualización

Una vez que los datos fueron introducidos a Elasticsearch, Kibana puede extraerlos con el objetivo de visualizar, analizar y obtener información relevante relacionado a

la seguridad de la red. Primero será necesario configurar un patrón de índice para acceder a los datos de Elasticsearch, para lo que será necesario abrir Kibana en la dirección `http://192.168.1.180:5601`. Dos pasos son necesario para extraer la información desde Elasticsearch:

- Proporcionar los patrones del índice: como se mencionó en el punto anterior los índices tienen un patrón en el nombre tal como se observa en la tabla 6. Se debe crear un patrón de índice para visualizar los índices generados por los tres Agentes. En la figura 13 se observa la creación del patrón del indicie mediante el uso del comodín * para que coinciden con todos los índices generados por el agente Filebeat.

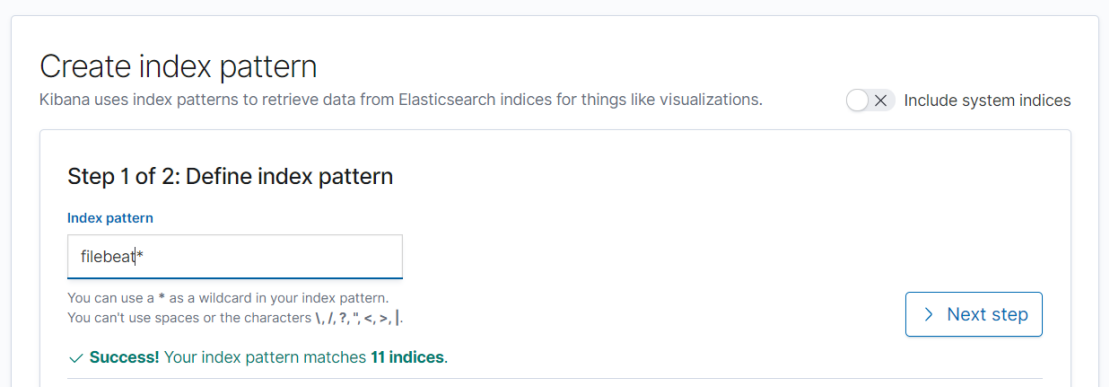


Figura 13: Creación de patrón de índice en kibana

- Seleccionar la marca de tiempo (timestamp), como se indica en la figura 14.

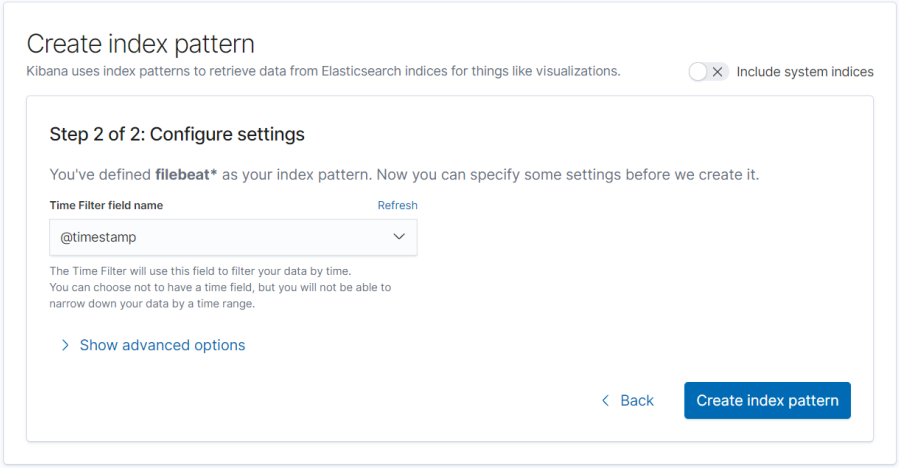


Figura 14: Seleccion de marca de tiempo en kibana

Después de realizar los dos pasos anteriores, se puede visualizar el nombre y campos del índice, así como los tipos de datos y detalles adicionales. (ver figura 15). Esto se debe realizar para recabar los datos de los indices con el formato:

- `filebeat-yyyy.MM.dd`
- `metricbeat-7.2.0-yyyy.MM.dd`
- `auditbeat-7.2.0-yyyy.MM.dd`

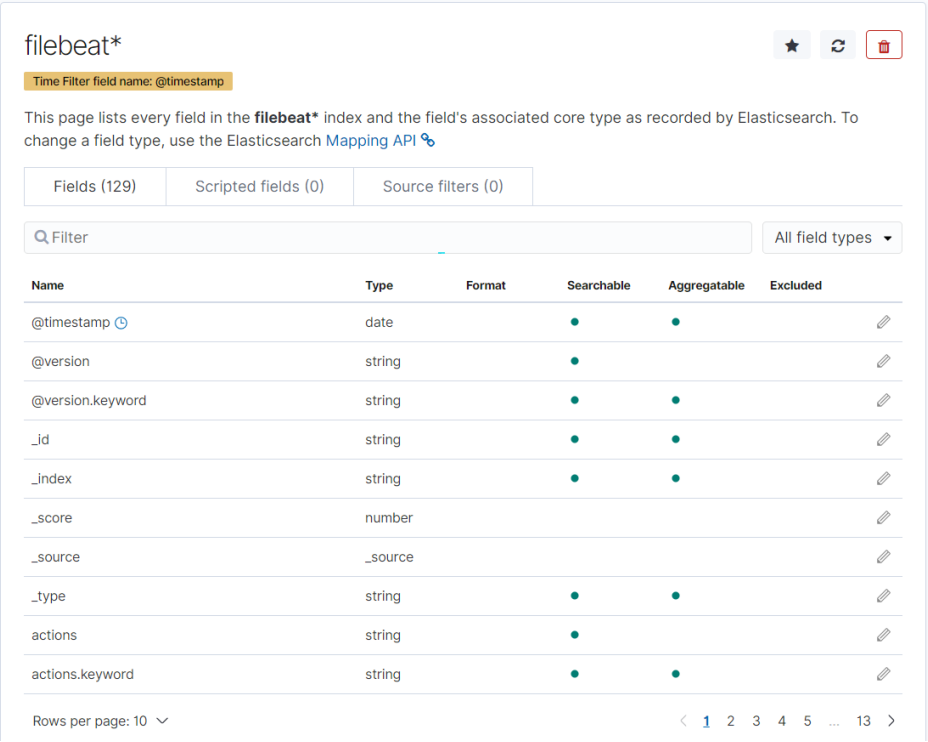


Figura 15: Campos del índice "filebeat"

La opción "*Discover*" dentro de Kibana permite realizar la exploración de los datos agregados, tal como se indica en la figura 16. En esta figura se puede observar un histograma, campos y valores de los datos ordenados de forma descendente en base a la marca de tiempo.

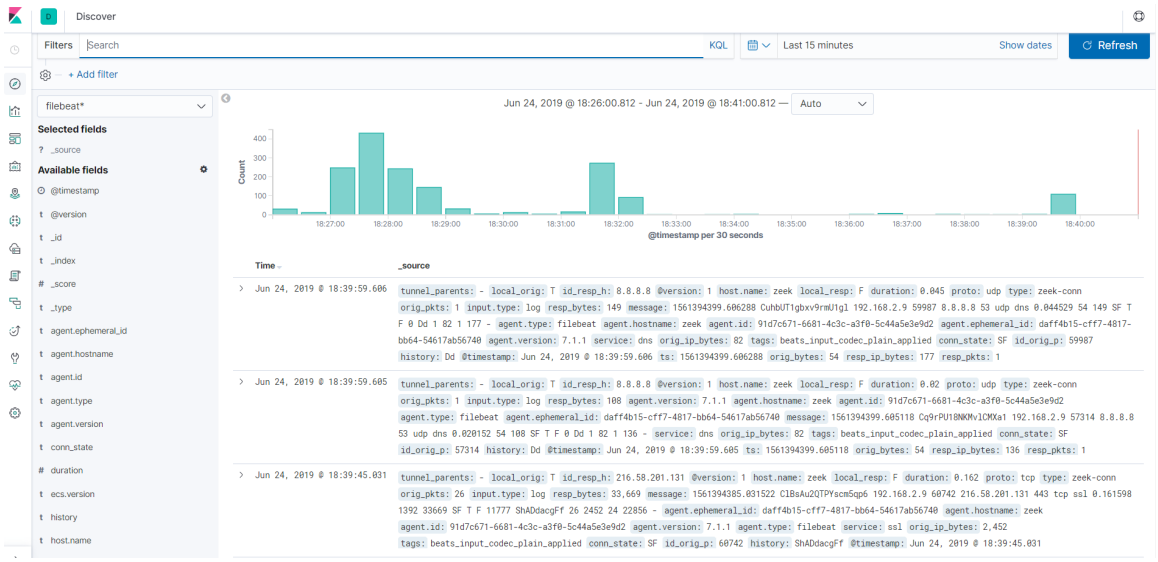


Figura 16: Opción "Discover" en kibana

Existe la opción de obtener detalles completos del conjunto de datos en formato tabla y en JSON con la posibilidad de aplicar filtros. (ver figura 17).

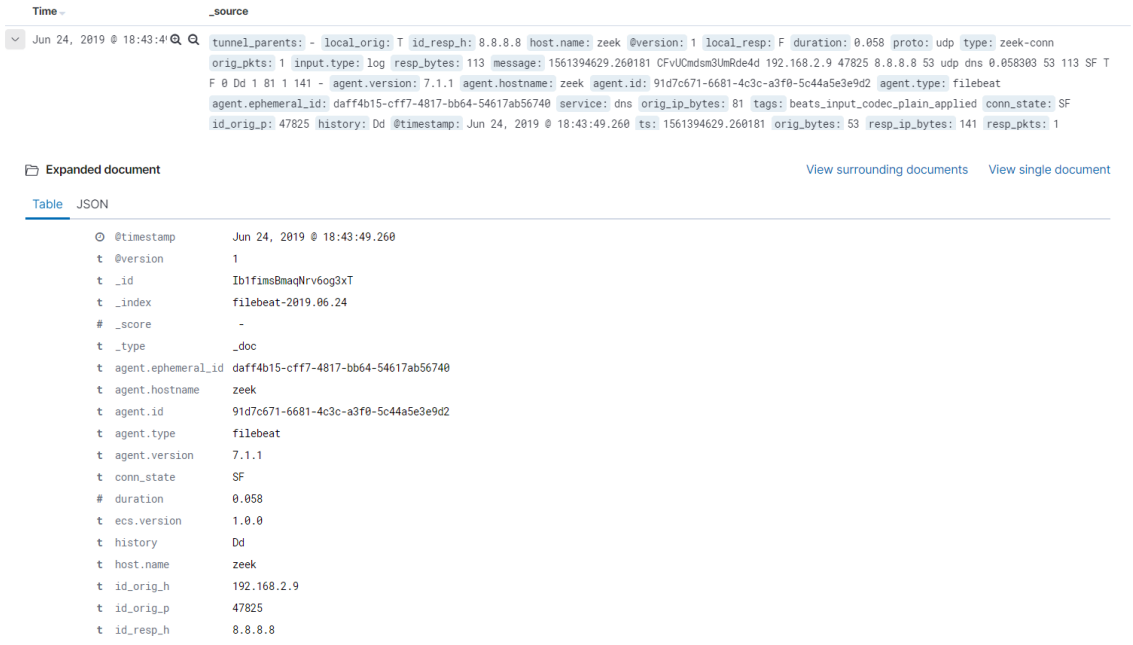


Figura 17: Documento en formato tabla

Los filtros permiten una mejor exploración de los datos para realizar búsquedas especificas con un determinado tipo de campo y valor utilizando. La búsqueda puede contener múltiples parámetros mediante el uso de operadores lógicos (and y or). En la figura 18 se realiza la búsqueda con el objetivo de obtener conexiones originadas desde una determina dirección IP (192.168.2.10) hacia un puerto en particular a partir de los datos recopilados del registro `conn.log` de Zeek.

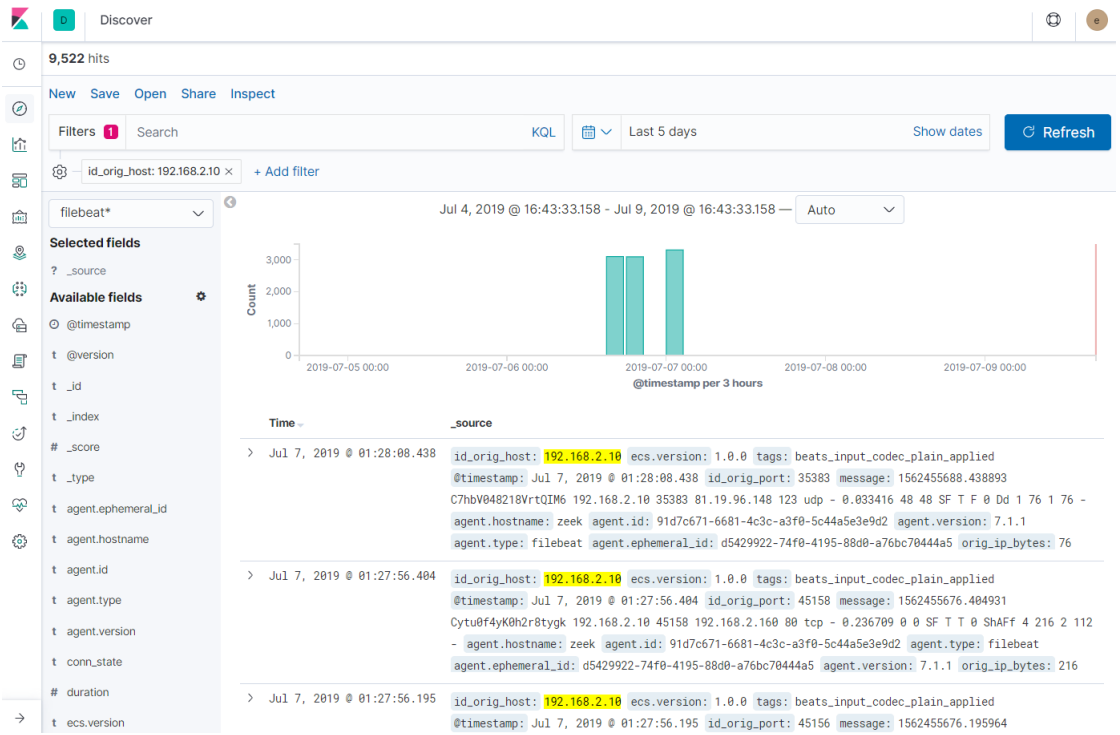


Figura 18: Aplicación de filtro en kibana

Una de las ventajas que ofrece kibana es la visualización de los datos en sus diversos tipos de diagramas tales como histogramas, diagrama circulares, mapas de calor, tablas, etc. Por ejemplo la figura 19 indica un diagrama de barras en donde se indica todas la conexiones originadas desde un determinado host con una dirección IP dentro de un periodo de tiempo.

Además es posible la creaciones de paneles (dashboards) en los cuales se agregan las diferentes visualizaciones creadas con el fin de proporcionar un rápido panorama de los eventos que se están generando o eventos anteriores.

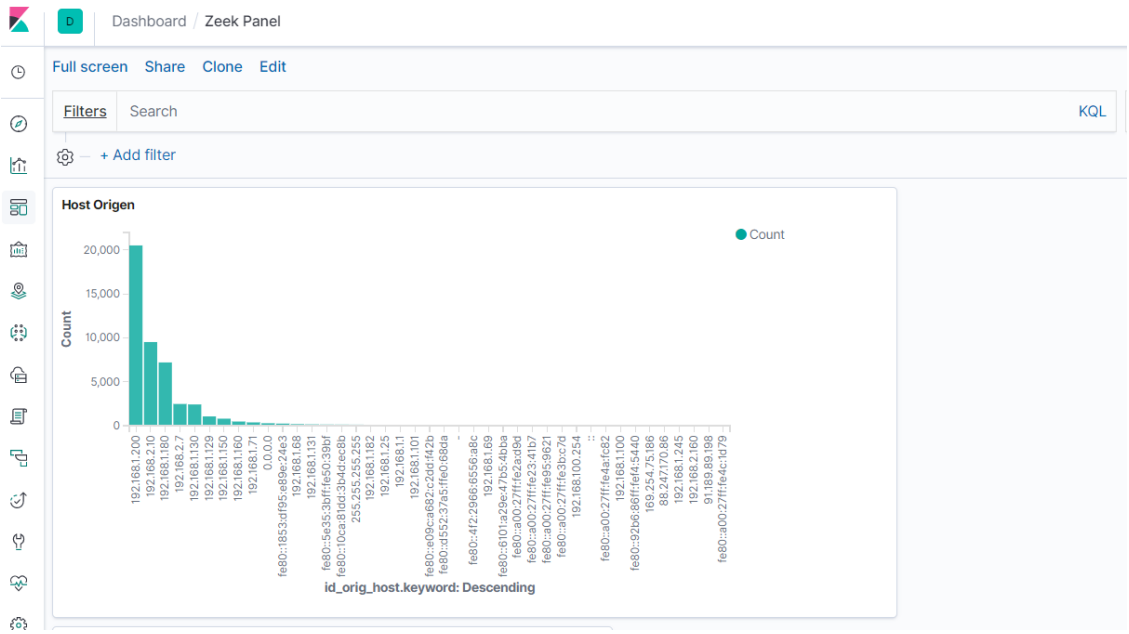


Figura 19: Diagrama de barras en kibana

9.6 Monitorización de Elastic Stack

La extensión denominada X-Pack posee la funcionalidad de monitorizar los servicios que conforman Elastic Stack. Aunque las características básicas de monitorización están presente en Elastic Stack, X-Pack proporciona capacidades adicionales del estado o funcionamiento de los diversos componentes de forma gratuita.

Entre las funcionalidades de monirotizacion que posee X-Pack están:

- Capacidad y uso de disco duro de host
- Cantidad de documentos e indices
- Monitorización del pipeline de Logstash
- Peticiones y tiempo de respuesta en KIabana

En kibana, la opción ”Stack Monitoring” muestra la pantalla principal con tres cuadros en donde se puede visualizar detalles del estado de Elasticsearck, Kibana y Logstash.

En la figura 20 se observa el cuadro con información relacionada a Elastcisearch sobre la versión, tiempo de actividad, numero de nodos que incluye el espacio total

de disco disponible, numero de indices junto con información sobre los documentos, uso de disco y otros datos.

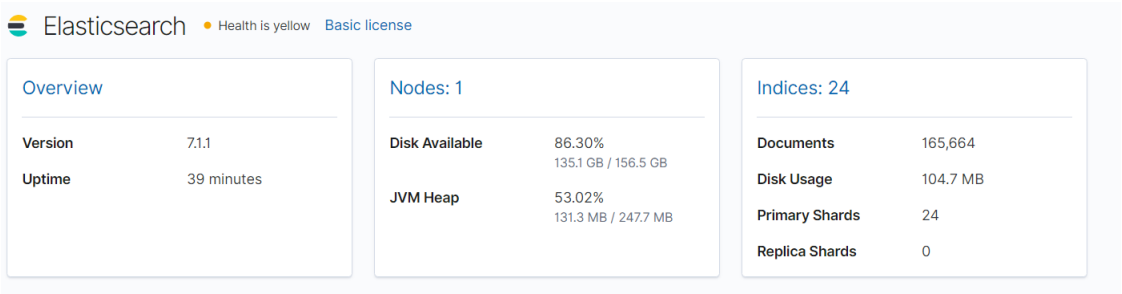


Figura 20: Información estadística de Elasticsearch

El cuadro de la información de Kibana se indica en la figura 21 indica información sobre su estado, el número de instancias de Kibana en funcionamiento, junto con el total de solicitudes, tiempo máximo de respuesta, conexiones y porcentaje de memoria utilizada

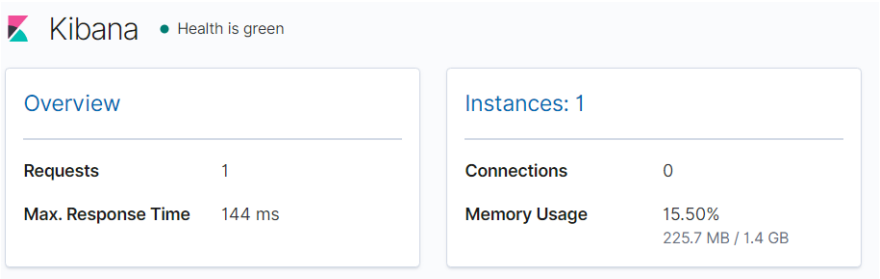


Figura 21: Información estadística de Kibana

Mientras que el cuadro de información para Logststash, muestra información básica sobre los eventos, nodos y pipeline, junto con cantidad de eventos emitidos/recibidos, disponibilidad y configuración de colas. La figura 22 se observa las estadísticas relacionadas a kibana.

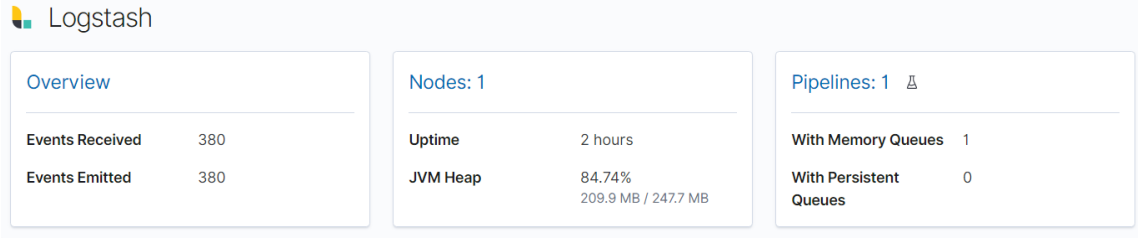


Figura 22: Información estadística de Logstash

Es posible obtener mayor detalle de la información para cada servicio. En la sección "Pipeline" en Logtash, se puede identificar el funcionamiento en tempo real de los diferentes elementos que componene el pipeline tales como Entradas (inputs), filtros (filters) y salidas (outputs). En el componente Filters se puede verificar la cantidad de eventos procesados por cada uno de los filtros creados para cada fichero que ingresa a Logsatsh.

Figura 23: Información estadística de Logstash

Adicionalmente, la figura 24 indica el estado de los indices que posee Elasticsearch, así como cantidad de documentos y tamaño en disco relacionados a estos.

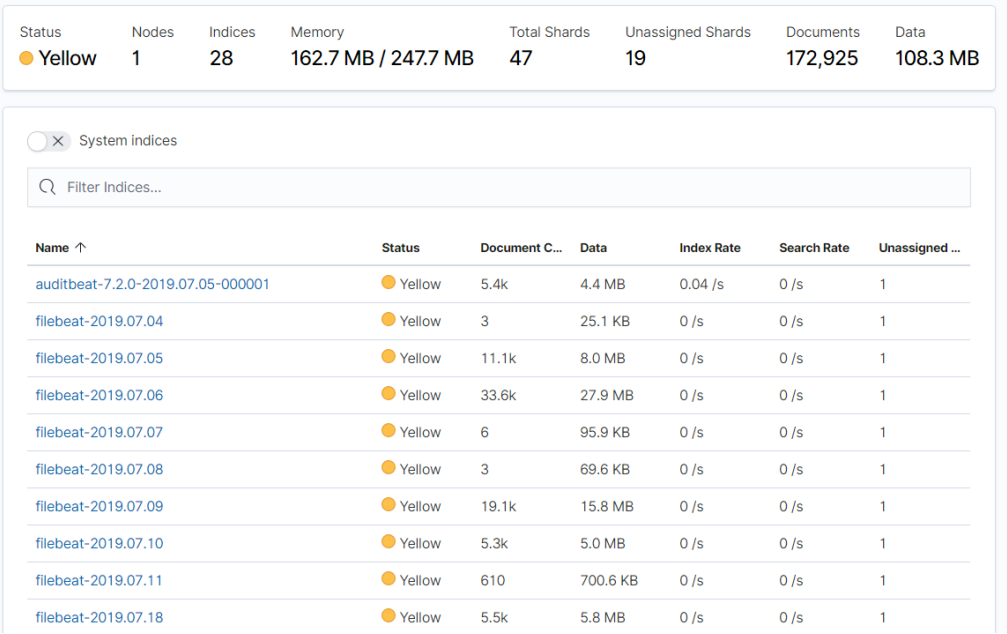


Figura 24: Estado de indices en Elastcisearch

Además, se puede monitorizar de forma gráfica la cantidad de eventos recibidos, emitidos y latencia durante un periodo de tiempo tal como se observa en la figura

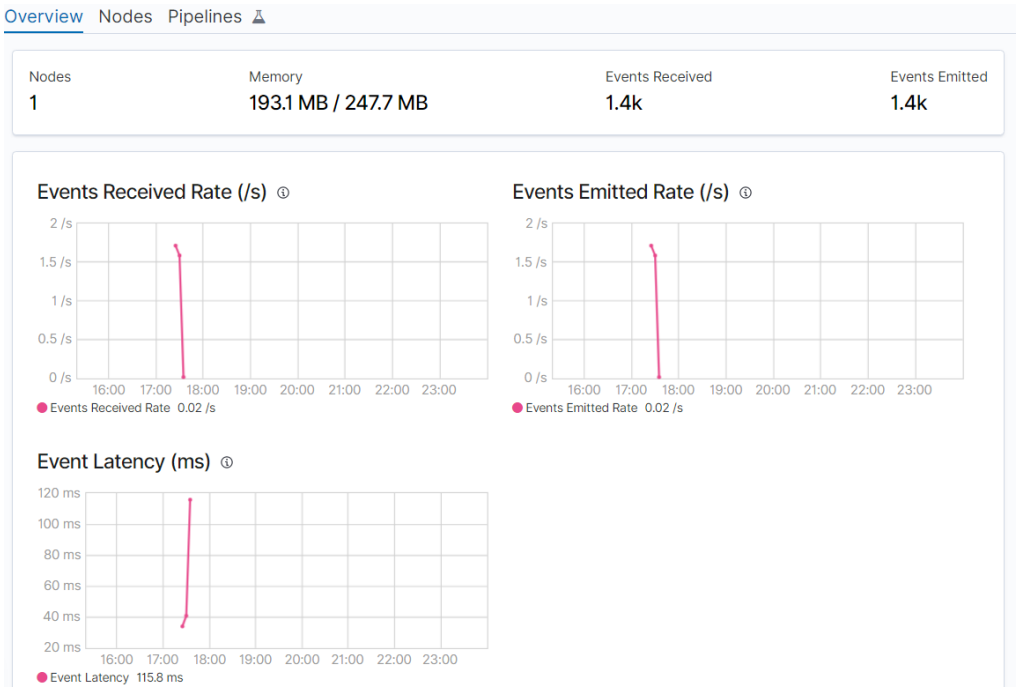


Figura 25: Información estadística de Logstash

9.7 Pruebas y resultados

9.7.1 Escenario

La comprobación del correcto funcionamiento del sistema de gestión de eventos se realiza mediante el despliegue de un escenario básico de pruebas en un entorno virtual como se observa en la figura 26.

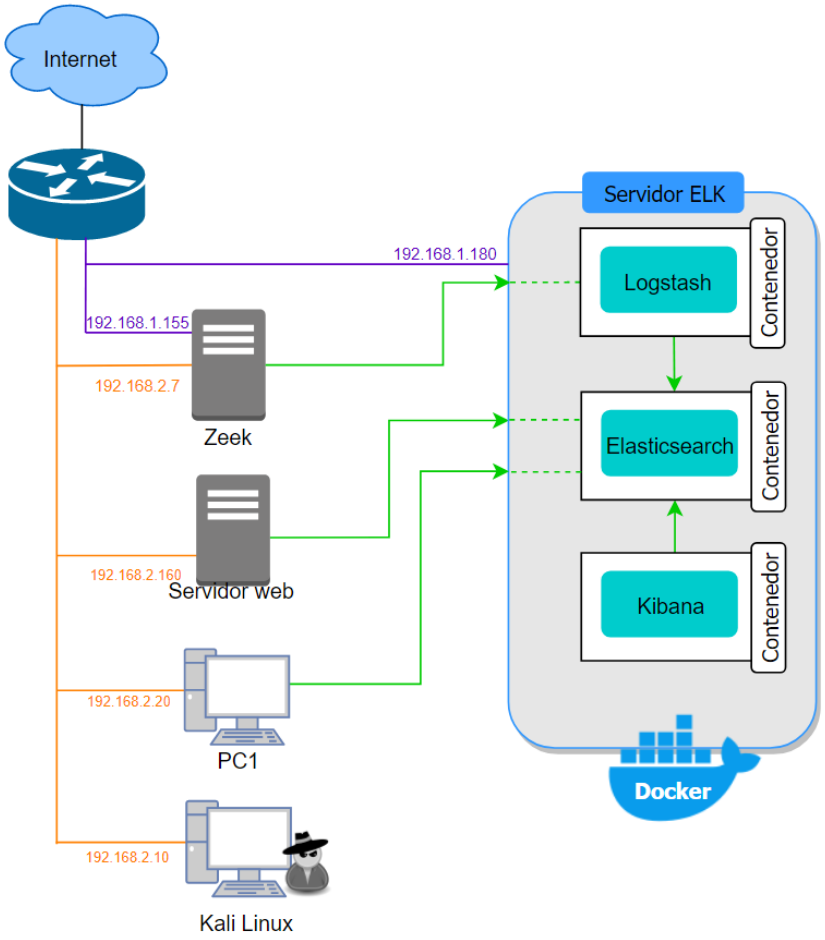


Figura 26: Escenario de pruebas de funcionamiento

Las pruebas de funcionamiento se basan principalmente en la generación de anomalías en base al análisis del tráfico realizado por la herramienta Zeek. Para lo que será necesario generar tráfico malicioso mediante y la simulacion de ataques por parte de Kali Linux.

Las actividades necesarias para llevar a cabo las pruebas de funcionamiento son las siguientes:

1. **Generación de tráfico malicioso:** para el caso del Framework de inteligencia, su correcto funcionamiento se basa en la correcta detección de tráfico malicioso generado desde el PC1. Cabe mencionar que el CIF desplegado consta de URLs maliciosas, por lo tanto basta con apuntar a una de estas en un navegador web para verificar la correcta detección.

2. **Reconocimiento y escaneo:** enumeración de servicios y escaneo de puertos mediante kali Linux y en donde el activo objetivo o víctima es el servidor web.

3. **Explotación:** vectores de ataques y lanzamiento de exploits, igualmente realizado a traves de Kali Linux y el objetivo o victima el servidor web.

Zeek debe ser capaz de detectar la actividad anómala en la red y generar logs para el análisis y visualización en Kibana. Así mismo el sistema de gestión de eventos debe recopilar datos de auditoría y estadísticos provenientes del servidor web y PC1. No se requiere ninguna acción por realizar mas que la correcta visualización de estos datos en la herramienta kibana.

9.7.2 Detección de anomalías

Intelligence Framework

Como se mencionó anteriormente, Zeek puede consumir datos de inteligencia generados desde diferentes fuentes para la detección de amenazas. En este caso se ha utilizado *feeds* disponibles en el proyecto CSIRT Gadgets. Al momento de producirse alguna coincidencia en el tráfico analizado por Zeek con cualquier indicador se genera eventos en el fichero `intel.log` tal como se observan a continuación

```
#fields ts uid id.orig_h id.orig_p id.resp_h id.resp_p seen.indicator ↩
    ↪ seen.indicator_type seen.where seen.node matched sources fuid ↩
    ↪ file_mime_type file_desc
#types time string addr port addr port string enum enum string set[enum↩
    ↪ ] set[string] string string string
1563659307.803980 CmyyuKjNqTCSDAXC5 192.168.2.20 60266 210.253.231.78 ↩
    ↪ 80 210.253.231.78 Intel::ADDR Conn::IN_RESP bro Intel::ADDR ↩
    ↪ csirtg.io - - -
1563659310.171801 C4jShj3Z2aFBmDOlOf 192.168.2.20 60268 210.253.231.78 ↩
    ↪ 80 210.253.231.78 Intel::ADDR Conn::IN_RESP bro Intel::ADDR ↩
    ↪ csirtg.io - - -
1563659310.187911 CBGP4w2rB4CQhNy0y9 192.168.2.20 60270 210.253.231.78 ↩
    ↪ 80 210.253.231.78 Intel::ADDR Conn::IN_RESP bro Intel::ADDR ↩
    ↪ csirtg.io - - -
```

```
1563659538.791922 CBn18h4XVPTFAnOy99 192.168.2.20 40874 14.160.39.30 80
    14.160.39.30 Intel::ADDR Conn::IN_RESP bro Intel::ADDR csirtg.
    io - - -
1563659539.031915 CnY86g40LdtHspN7x7 192.168.2.20 40876 14.160.39.30 80
    14.160.39.30 Intel::ADDR Conn::IN_RESP bro Intel::ADDR csirtg.
    io - - -
1563659542.635950 CuAWu21zHKEYzkghF5 192.168.2.20 40884 14.160.39.30 80
    14.160.39.30 Intel::ADDR Conn::IN_RESP bro Intel::ADDR csirtg.
    io - - -
```

En este caso los indicadores corresponden a direcciones IP maliciosas y el tráfico es originado desde el PC1. Ademas, la informacion acerca de esta anomalía es reportada en el registro `notice.log`.

Enumeración y escaneo de puertos

La herramienta Nmap en Kali Linux se emplea para realizar la enumeración de servicio y escaneo de puertos. Zeek debe ser capaz de detectar esta actividad y generar información de la anomalía en el registro notice.log. En la siguiente salida se indica la información generada por parte de Zeek en el registro notice.log cuando se produce un escaneo de puertos en la red.

```
#fields ts uid id.orig_h id.orig_p id.resp_h id.resp_p fuid
    file_mime_type file_desc proto note msg sub src dst p n
    peer_descr actions suppress_for ...
#types time string addr port addr port string string string enum enum
    string string addr addr port ...
1562422852.883355 - - - - - - - -- Scan::Port_Scan 192.168.2.10 scanned
    at least 15 unique ports of host 192.168.2.160 in 0m0s local
    192.168.2.10 192.168.2.160 - - -Notice::ACTION_LOG 3600.000000 F
```

De la salida anterior se puede establecer que fue realizado un escaneo de puerto desde la dirección IP 192.168.2.10 hacia el equipo con dirección IP 192.168.2.160.

Incluso Zeek detecta escaneos de puertos cuando Nmap usa mecanismos para evitar la detección de IDs mediante la reducción de la velocidad del escaneo. La siguiente salida indica la detección de un escaneo utilizando la opción T1 de Nmap, donde puede tardar hasta 5 segundos para establece cada conexión y evitar así la detección. Se puede apreciar que el tiempo de duración de conexión es diferente a la salida anterior, en este caso el valor es de 3m45s.

```
...
```

```
1562690949.464197 - - - - - - - -- Scan::Port_Scan 192.168.2.10 scanned
    ↪  at least 15 unique ports of host 192.168.2.160 in 3m45s local
    ↪ 192.168.2.10 192.168.2.160 - - -Notice::ACTION_LOG 3600.000000 F
    ↪  - - - - -
```

SQL Injection

Zeek detecta posibles ataques de inyección SQL mediante el script denominado detect-sqli.zeek. Igual que el caso anterior se utiliza Kali Linux para realizar esta explotación hacia el servidor web. La herramienta usada es *sqlmap* que se emplea para la detección y explotación de vulnerabilidades SQL injection.

El log generado por Zeek determina tanto la maquina víctima como el atacante como se observa a continuación

```
#fields ts uid id.orig_h id.orig_p id.resp_h id.resp_p fuid
    ↪ file_mime_type file_desc proto note msg sub src dst p n
    ↪ peer_descr actions suppress_for dropped remote_location.
    ↪ country_code remote_location.region remote_location.city
    ↪ remote_location.latitude remote_location.longitude
#types time string addr port addr port string string string enum enum
    ↪ string string addr addr port count string set[enum] interval
    ↪ bool string string string double double
1562454048.156867 - - - - - - - - - HTTP::SQL_Injection_Attacker An SQL
    ↪  injection attacker was discovered! - 192.168.2.10 - - - -
    ↪ Notice::ACTION_LOG 3600.000000 F - -- - -
1562454048.156867 - - - - - - - - - HTTP::SQL_Injection_Victim An SQL
    ↪ injection victim was discovered! - 192.168.2.160 -- - - Notice::
    ↪ ACTION_LOG 3600.000000 F - - - --
```

Fuerza Bruta

El script detect-bruteforcing.bro de Zeek permite la detección de ataques de fuerza en servicios ftp o ssh. Al igual que los casos anteriores el registro notice.log almacena información relevante acerca de la anomalía producida tal como se observa en la siguiente salida.

```
- SSH::Password_Guessing 192.168.2.10 appears to be guessing SSH
    ↪ passwords (seen in 30 connections). Sampled servers:
    ↪ 192.168.2.7, 192.168.2.7, 192.168.2.7, 192.168.2.7, 192.168.2.7
    ↪ 192.168.2.10 - - - -Notice::ACTION_LOG 3600.000000 F - - - - -
```

9.7.3 Visualización de eventos

Anomalías detectadas por Zeek

- Anomalías registradas en el fichero notice.log de Zeek

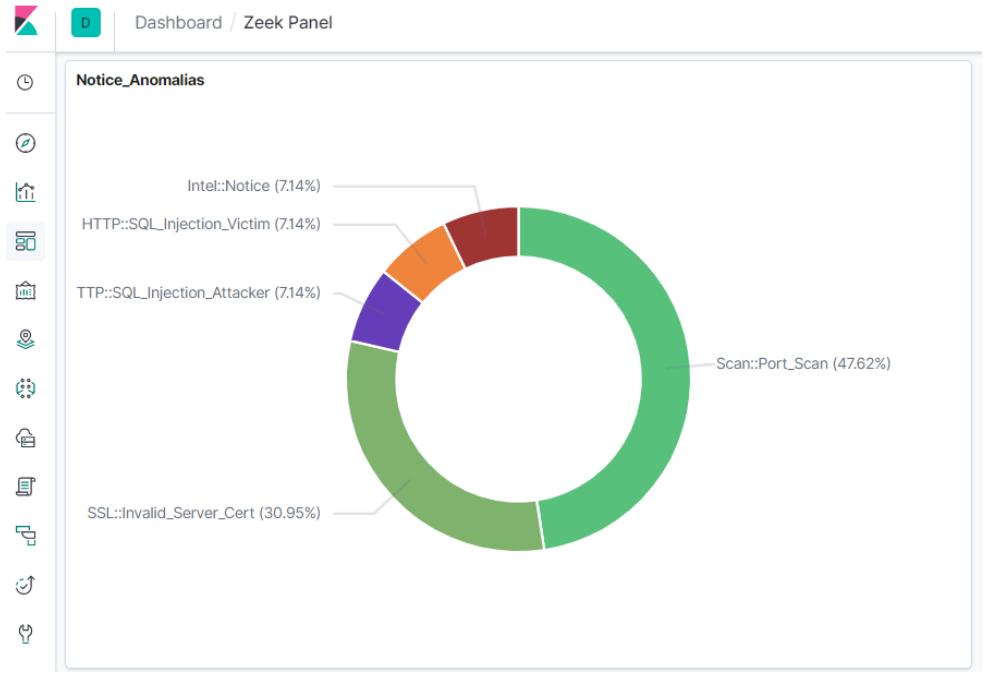


Figura 27: Anomalías de seguridad registradas

- Lista de direcciones IP origen de anomalías: *SQL Injection* y escaneo de puertos

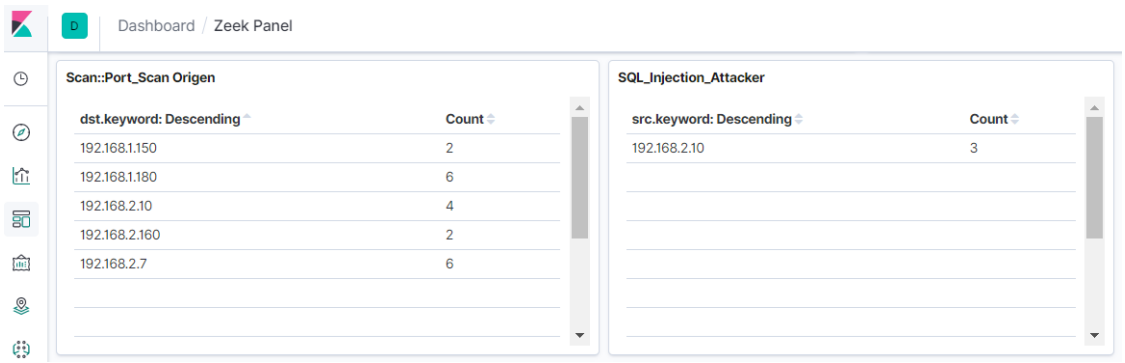


Figura 28: Dirección IP origen de anomalías

- Listado de direcciones IP origen de anomalías

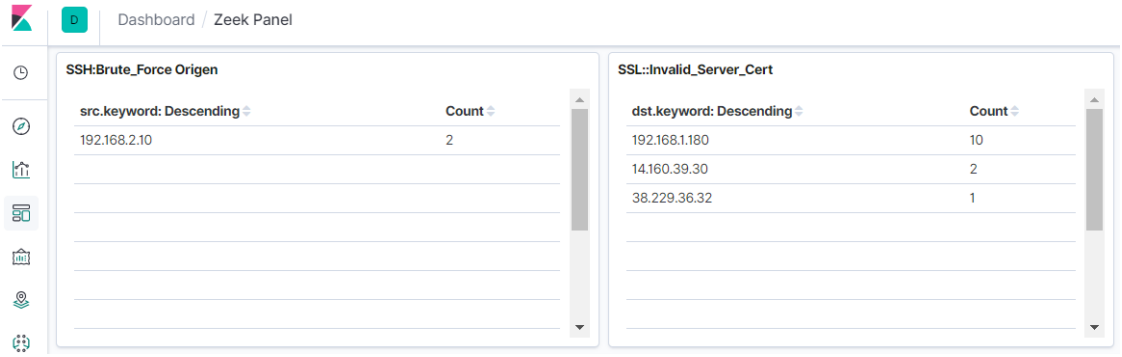


Figura 29: Anomalías de seguridad registradas

- Resumen de eventos registrado en el fichero notice.log

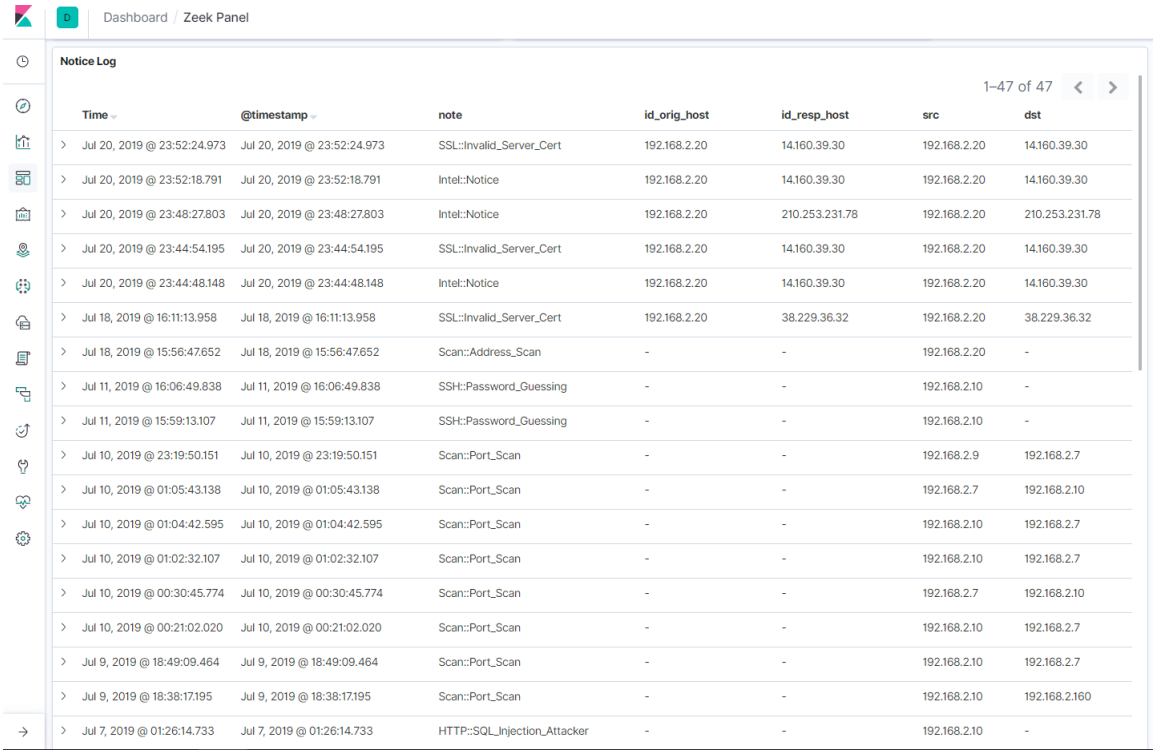


Figura 30: Lista de anomalías de seguridad registradas

- Eventos de seguridad generados por el Framework de inteligencia de Zeek, indicadores basados en IP y Dominio.

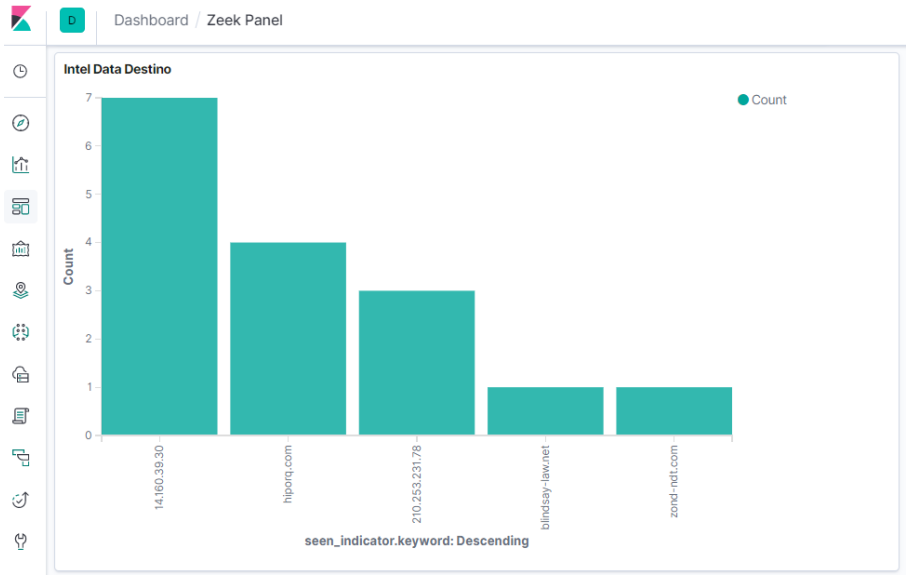


Figura 31: Anomalías de detectadas a través del Framework de inteligencia

Eventos de auditoría

- Información de Hosts activos monitorizados con Auditbeat

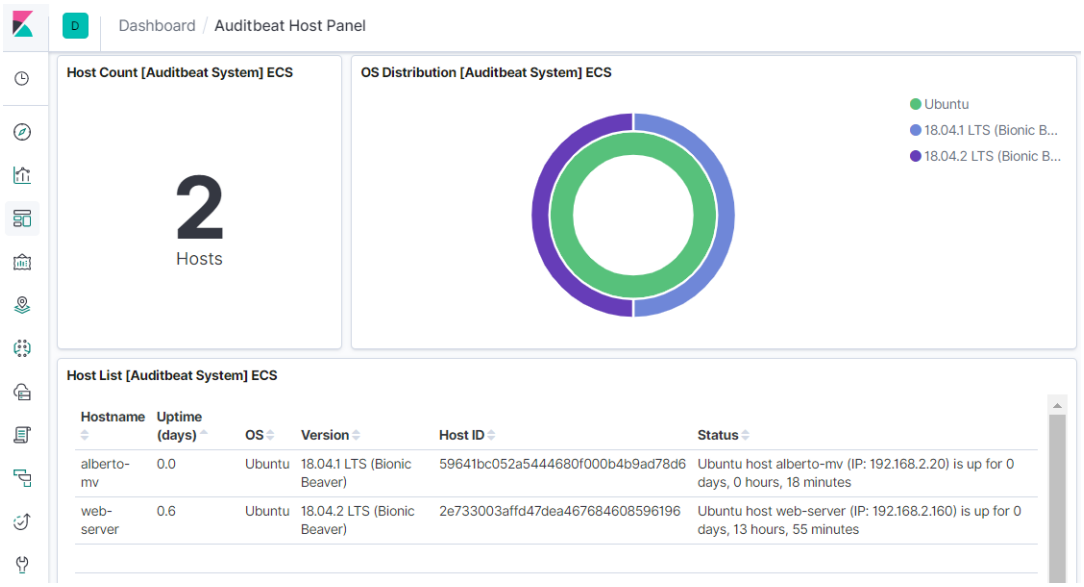


Figura 32: Monitorización de Host

- Eventos login de usuario: histograma, acciones, login exitosos y fallidos

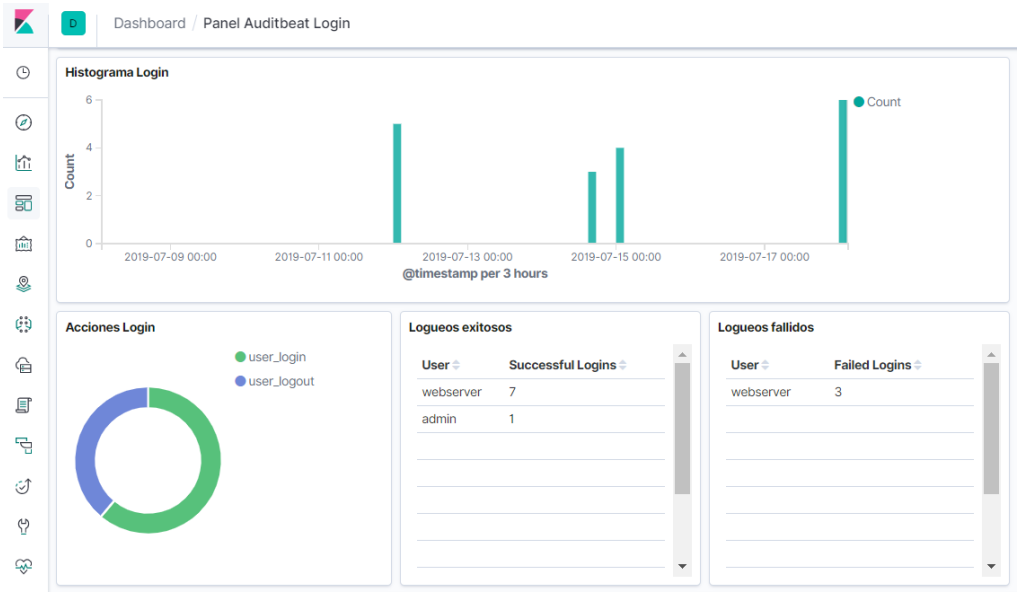


Figura 33: Eventos login de usuarios

- Resumen eventos login

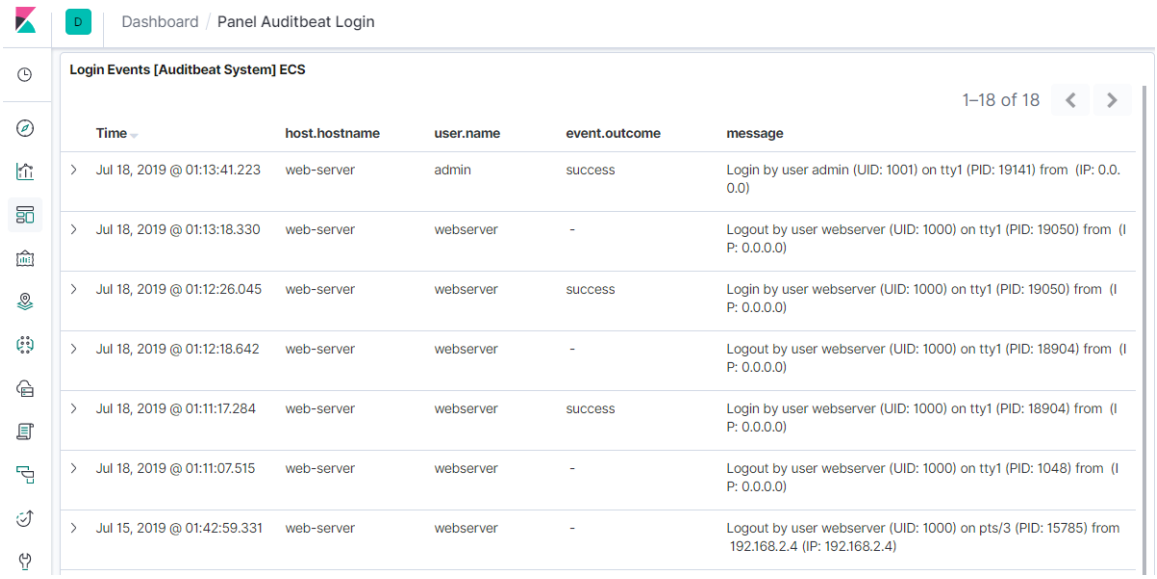


Figura 34: Resumen eventos login

- Información de integridad: usuarios propietarios y acciones

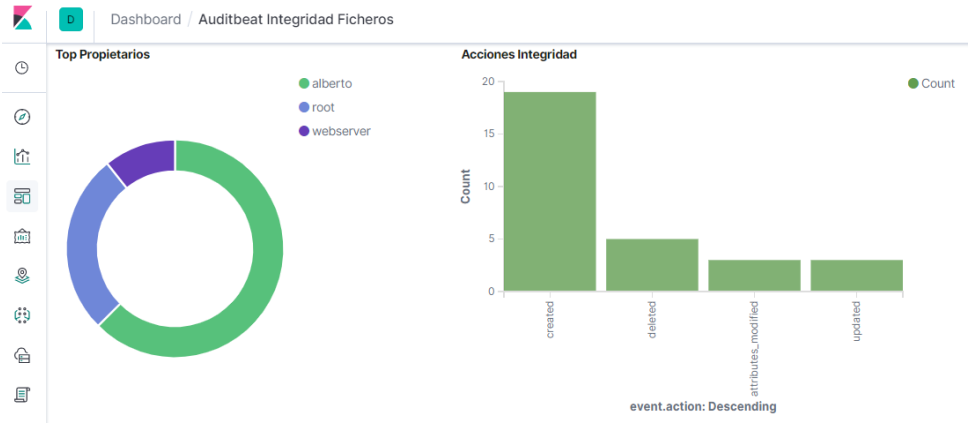


Figura 35: Información de integridad de ficheros y directorios

- Cantidad de eventos de integridad en un periodo de tiempo

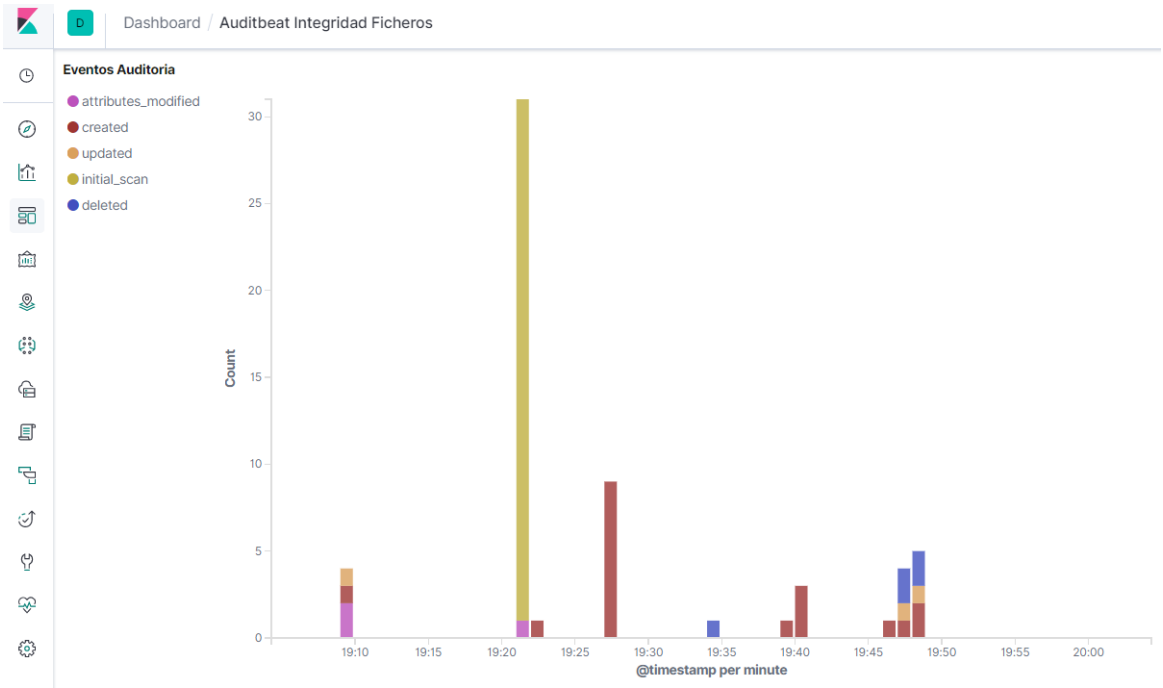


Figura 36: Resumen de eventos de integridad en el tiempo

- Resumen de eventos de integridad de ficheros y directorios

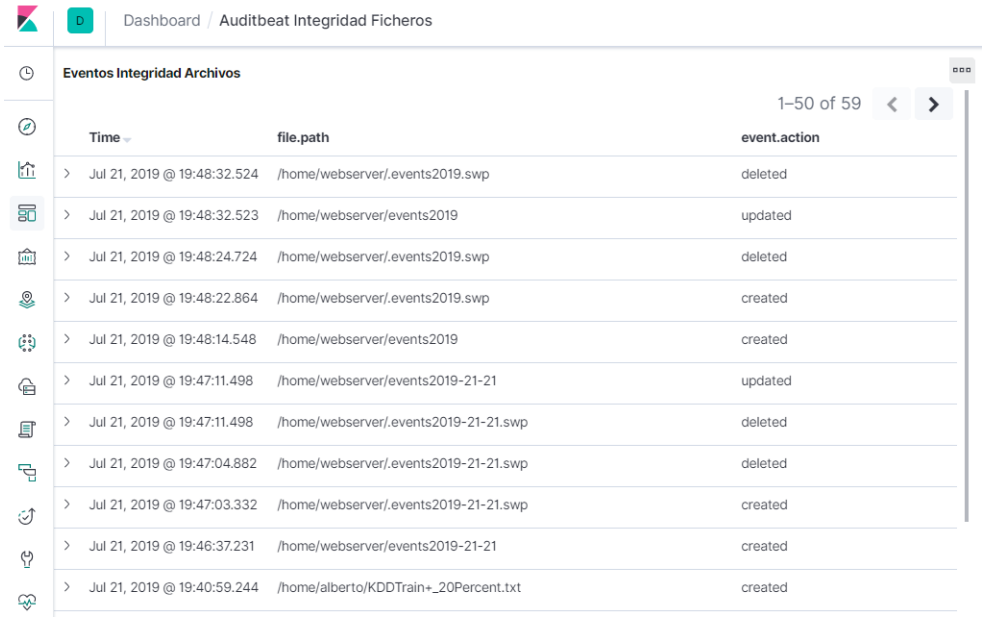


Figura 37: Resumen de eventos de integridad

- Total de eventos de integridad por Host

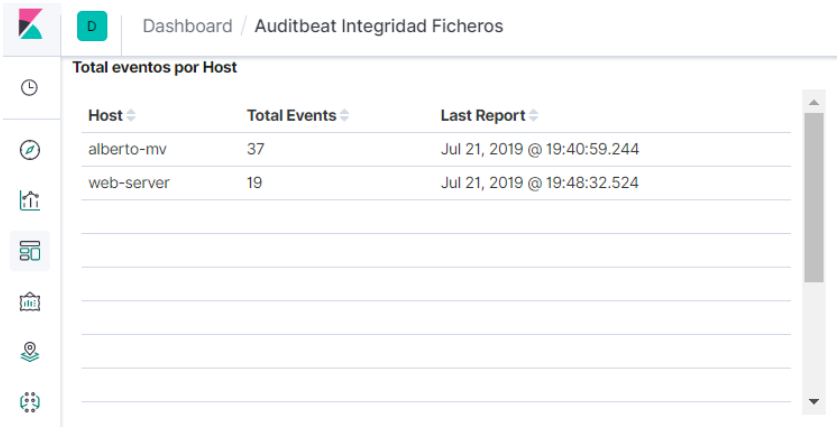


Figura 38: Total de eventos por Host

- Estado de procesos del sistema: iniciados, detenidos y erróneos

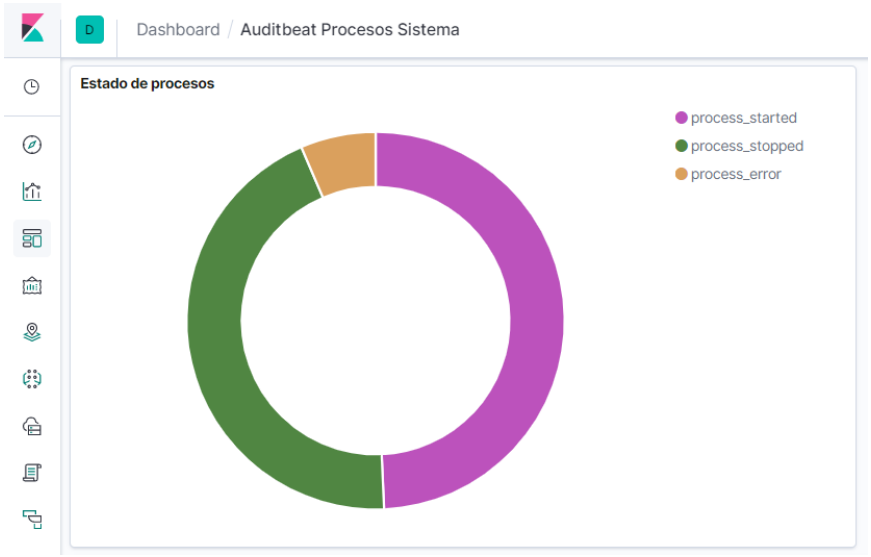
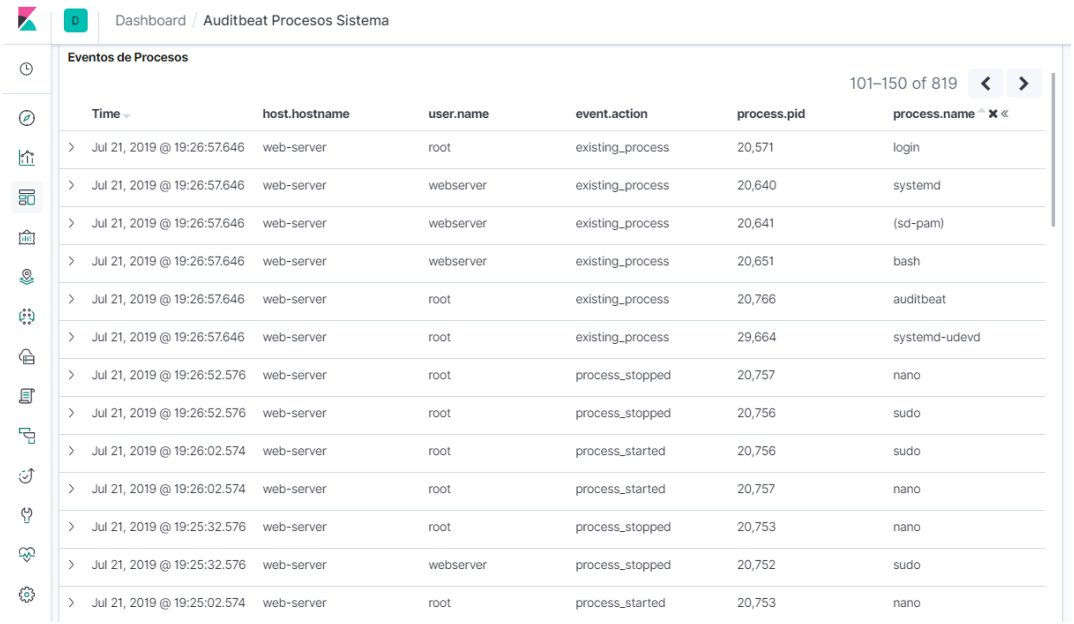


Figura 39: Estado de procesos del sistema

- Resumen de eventos de procesos de sistema



Time	host.hostname	user.name	event.action	process.pid	process.name
Jul 21, 2019 @ 19:26:57.646	web-server	root	existing_process	20,571	login
Jul 21, 2019 @ 19:26:57.646	web-server	webserver	existing_process	20,640	systemd
Jul 21, 2019 @ 19:26:57.646	web-server	webserver	existing_process	20,641	(sd-pam)
Jul 21, 2019 @ 19:26:57.646	web-server	webserver	existing_process	20,651	bash
Jul 21, 2019 @ 19:26:57.646	web-server	root	existing_process	20,766	auditbeat
Jul 21, 2019 @ 19:26:57.646	web-server	root	existing_process	29,664	systemd-udevd
Jul 21, 2019 @ 19:26:52.576	web-server	root	process_stopped	20,757	nano
Jul 21, 2019 @ 19:26:52.576	web-server	root	process_stopped	20,756	sudo
Jul 21, 2019 @ 19:26:02.574	web-server	root	process_started	20,756	sudo
Jul 21, 2019 @ 19:26:02.574	web-server	root	process_started	20,757	nano
Jul 21, 2019 @ 19:25:32.576	web-server	root	process_stopped	20,753	nano
Jul 21, 2019 @ 19:25:32.576	web-server	webserver	process_stopped	20,752	sudo
Jul 21, 2019 @ 19:25:02.574	web-server	root	process_started	20,753	nano

Figura 40: Resumen de eventos de procesos

Datos de contenido completo

- Cantidad de tramas por longitud

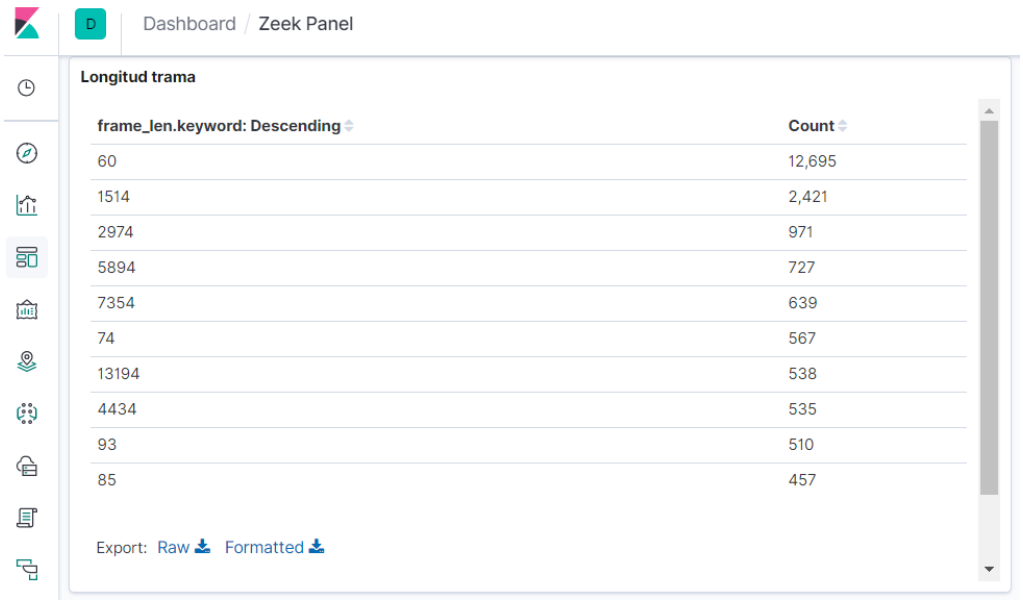


Figura 41: Longitud de tramas

- Cantidad de paquetes IP por longitud

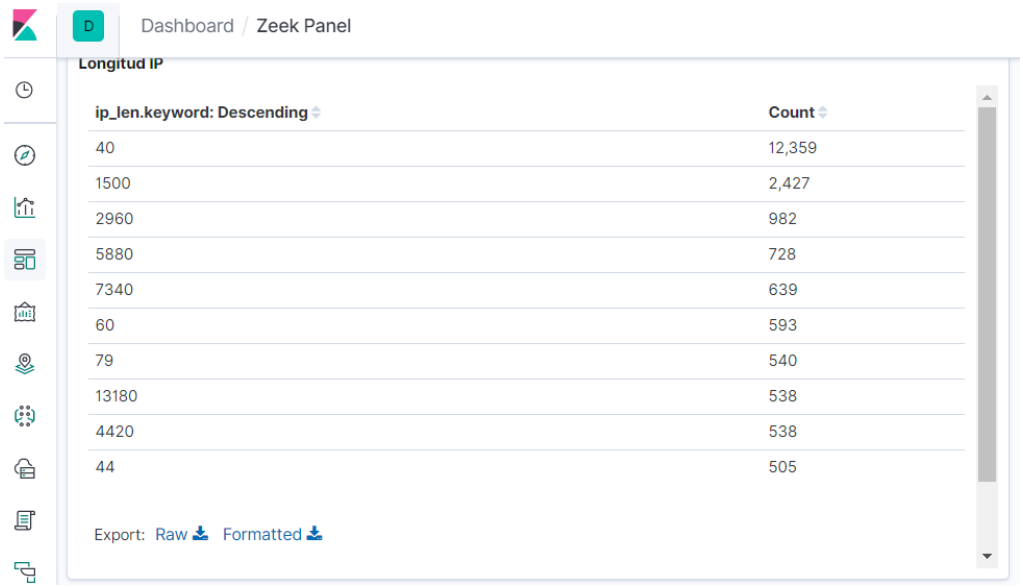


Figura 42: Longitud de paquetes IP

- Tramas por direcciones Ethernet destino

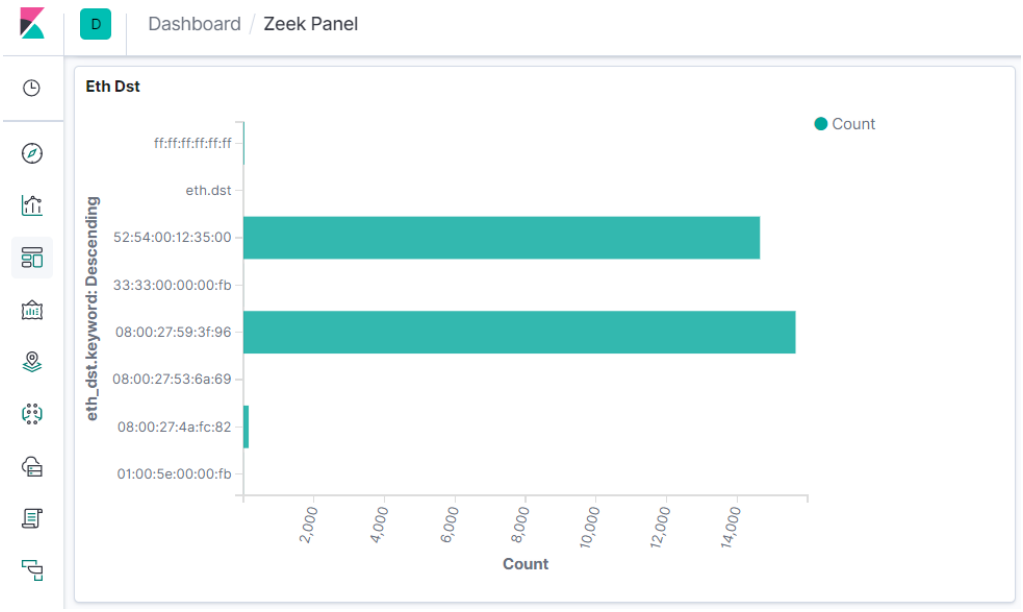


Figura 43: Direcciones Ethernet destino

- Tramas por direcciones Ethernet fuente

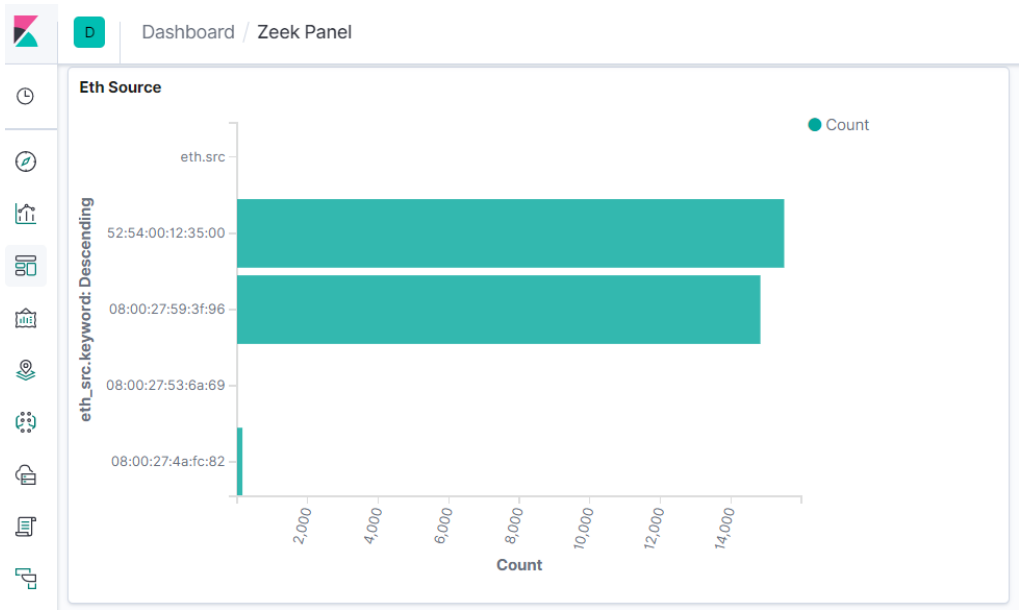


Figura 44: Direcciones Ethernet fuente

- Cantidad de paquetes por versión IP

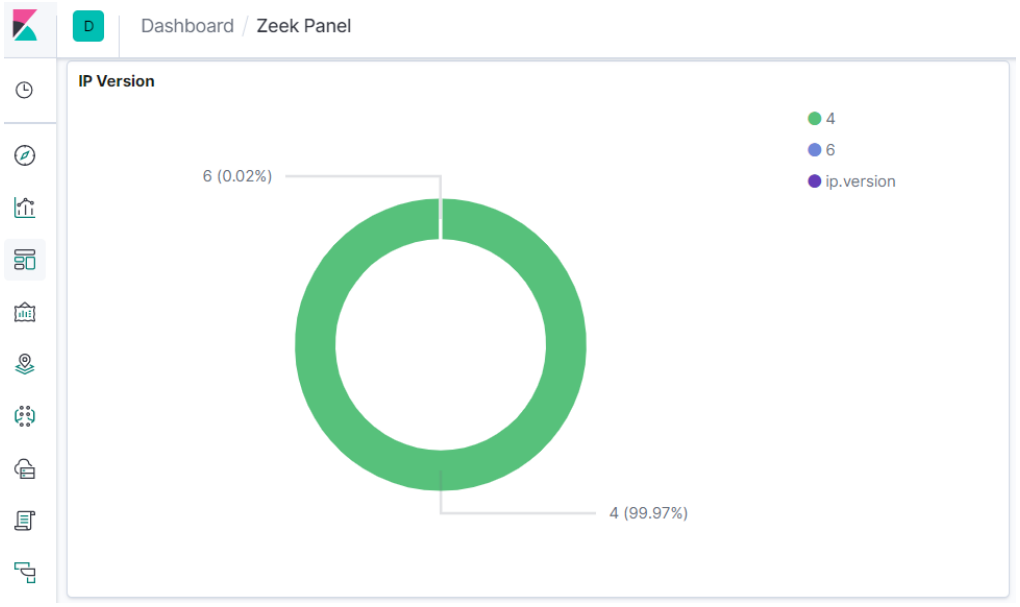


Figura 45: Versión IP

- Pila de protocolos de trama

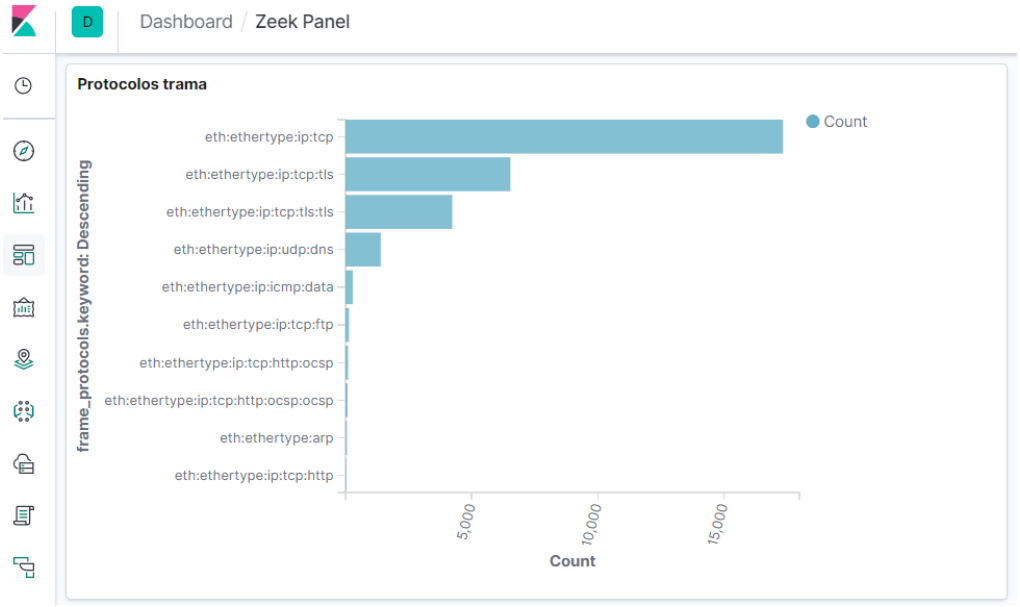


Figura 46: Pila de protocolos

Datos estadísticos

- Medidores de consumo: CPU, carga, memoria y disco

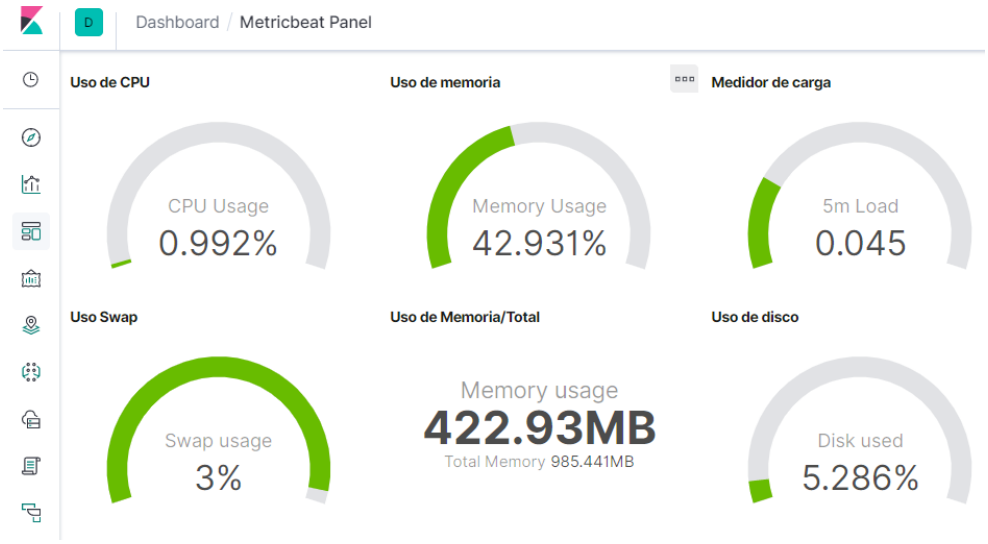


Figura 47: Medidores de métricas del sistema

- Uso de memoria del sistema

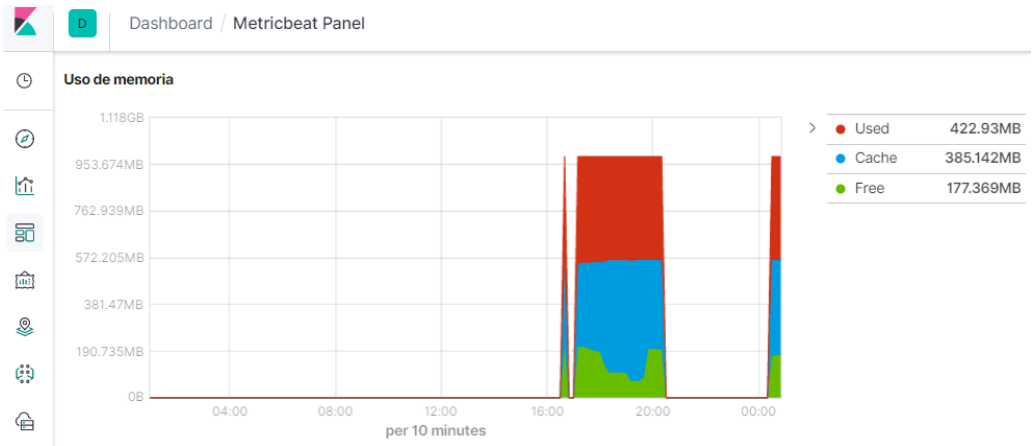


Figura 48: Uso de memoria

- Uso de CPU del sistema

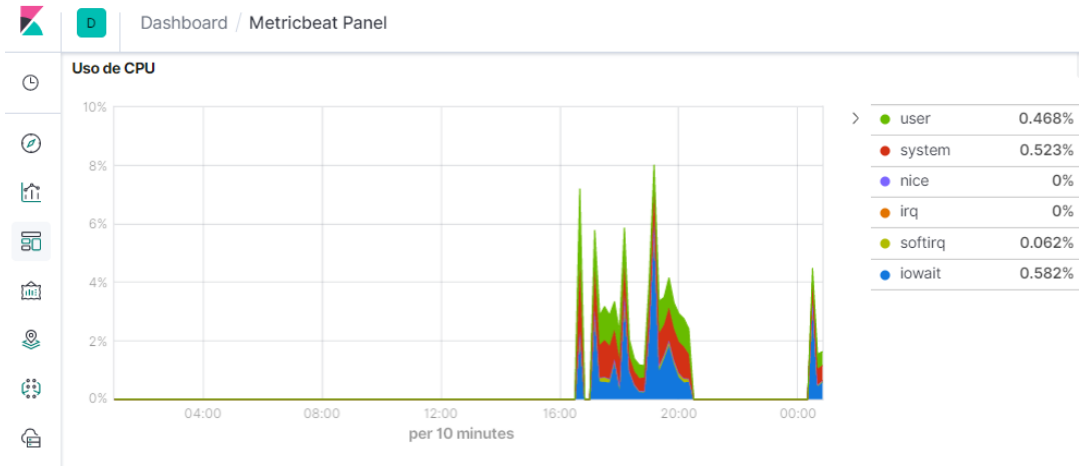


Figura 49: Uso de CPU

- Procesos por uso de memoria

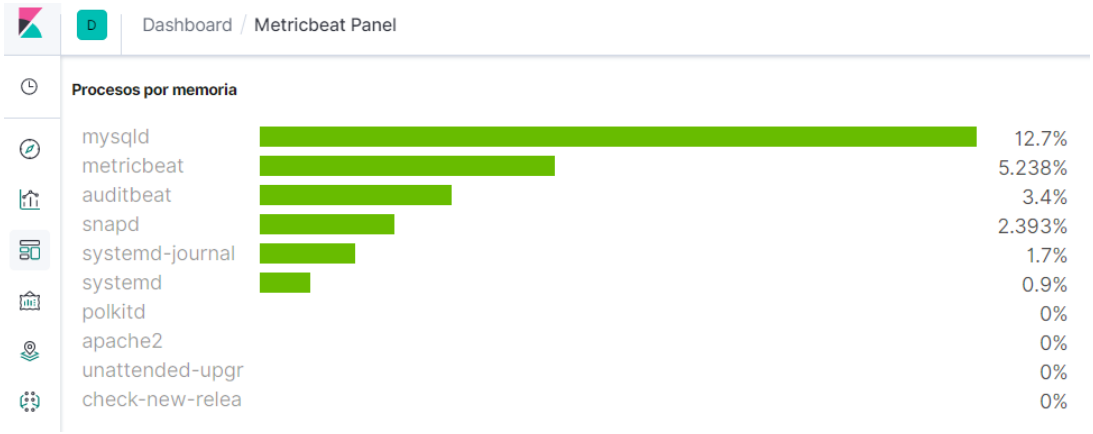


Figura 50: Top de procesos por memoria

- Procesos por uso de CPU

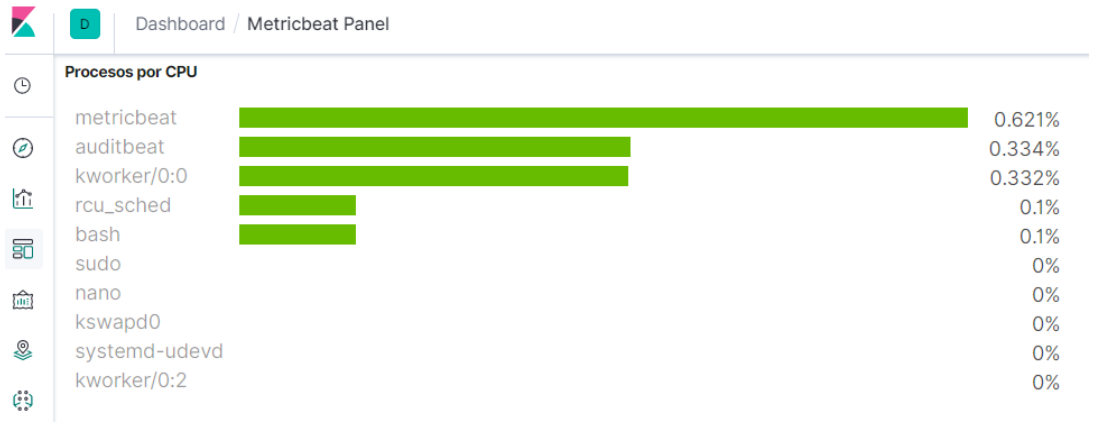


Figura 51: Top de procesos por CPU

- Tráfico de red (Bytes/segundo)

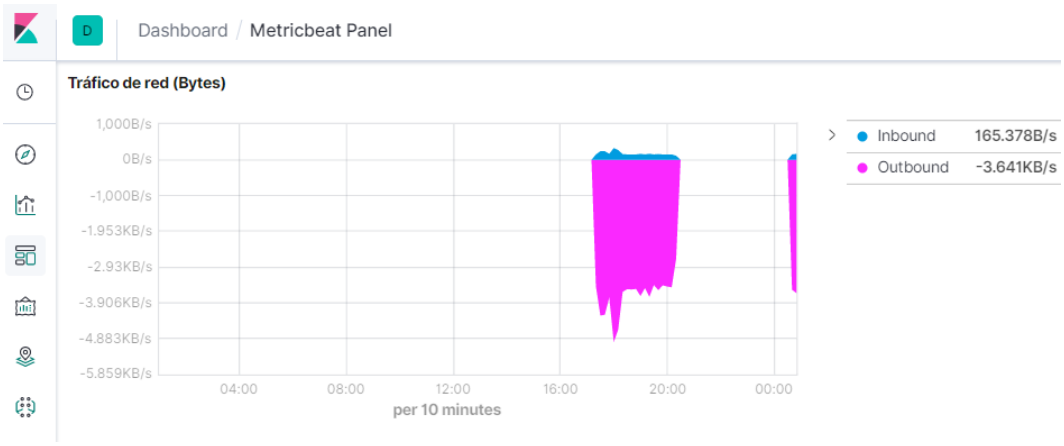


Figura 52: Tráfico de red

Las gráficas anteriores indican la gran capacidad que posee la solución para el análisis y visualización de eventos de seguridad de diferentes fuentes y tipos de datos. Aunque la solución es capaz de recopilar eventos de distintas fuentes, la principal fuente de información de eventos es el IDS Zeek el cual detecta anomalías en base un su lenguaje scripting. A manera de resumen, la cantidad de anomalías detectadas por Zeek se observan en la tabla 7.

Ítem	**Anomalía**	**Eventos**	**IPs/URLs maliciosas**
1	Escaneo de puertos	20	192.168.1.150 192.168.1.180 192.168.2.10 192.168.2.160 192.168.2.7
2	Certificados SSL inválidos	13	192.168.1.180 14.160.39.30 38.229.36.32
3	SQL injection	3	192.168.2.10
4	Detección por datos de inteligencia	3	14.160.39.30 hiporq.com 210.253.231.78 blindsay-law.net zond-ndt.com
5	Fuerza Bruta SSH	2	192.168.2.10

Tabla 7: Resumen de detección de anomalías

9.8 Planificación temporal

La siguiente figura muestra el cronograma del proyecto el cual determina la duración del presente proyecto en ejecutarse basado en un diagrama de Gantt.

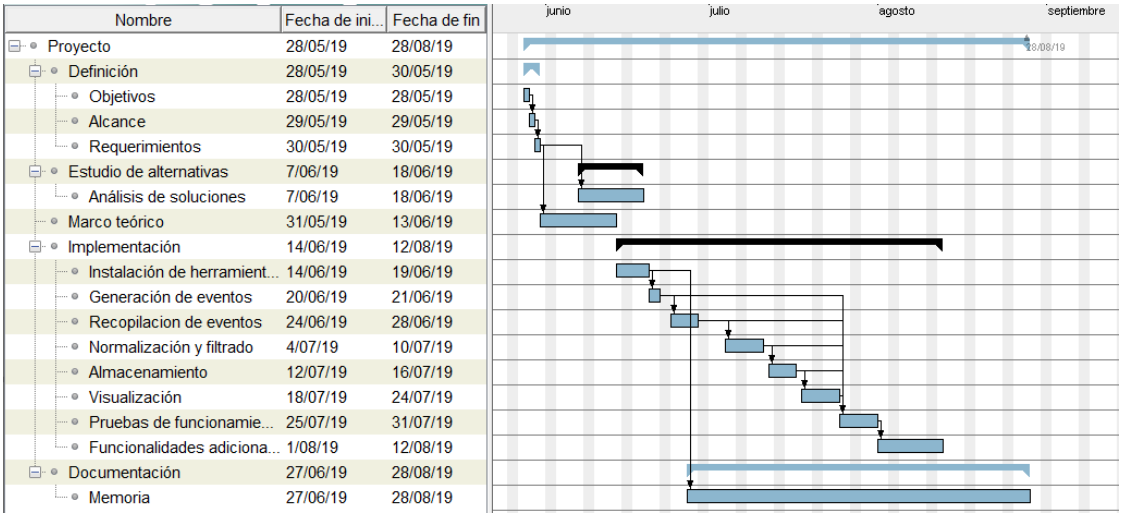


Figura 53: Planificación del proyecto

9.9 Resumen del presupuesto

Ítem	Descripción	Coste Total (€)
1	Equipamiento	4390.00
2	Mano de obra	2400.00
	Total	**6790.00**

Tabla 8: Resumen de presupuesto

9.10 Orden de prioridad de los documentos

1. Especificaciones
2. Memoria
3. Mediciones
4. Presupuesto
5. Marco Teórico
6. Anexos

SISTEMA DE GESTIÓN DE EVENTOS RELACIONADOS CON ANOMALÍAS EN LA SEGURIDAD EN UN ENTORNO OPEN SOURCE BASADOS EN DOCKER

MARCO TEÓRICO

Suministrador:
Alberto Mejía Viteri
alberto.mejiaviteri@alum.uca.es

Puerto Real, septiembre 2019

Marco teórico

1 Monitorización de seguridad de red

1.1 Colección

La colección de datos es la primera etapa de la monitorización de la seguridad de la red. Se basa en recopilación de información o datos necesarios para realizar tareas de análisis y detección para establecer la existencia de actividades normales o maliciosas. Las fuentes más comunes que son colectadas son: datos de contenido completo, datos de sesión, información estadística, alarmas y logs.

1.1.1 Datos de contenido completo

Consiste en la captura completa o parcial de paquetes de circulan por una determinada red principalmente a través de herramientas denominadas *sniffers*. La principal ventaja que ofrece la captura de paquetes es permitir un análisis con mayor detalle y profundidad de posibles incidentes o anomalías ocurridas en la red en comparación de otras datos recopilados.

Aunque estos datos proporcionan información completa, puede ser relativamente complejo almacenarlos durante un periodo de tiempo amplio debido a la gran cantidad de información capturada. Previo a efectuar el análisis, se requiere mecanismos que permitan una depuración de los datos con el fin de reducir la cantidad de estos. Por ejemplo, uno de los aspectos es no tomar en cuenta la recopilación de tráfico cifrado.

La principal librería para realizar la captura de tráfico es libpcap. Libpcap es una librería *open source* escrita en C/C++ y permite la portabilidad para diversos sistemas operativos. Existen dos versiones de esta librería para entornos Windows: WinPcap y Npcap. La primera es un versión desactualizada y cada vez más en desuso, mientras Npcap se basa en la última versión de libpcap y disponible para Windows 7 y versiones posteriores. Las principales herramientas conocidas para la captura de trafico son: tcpdump, Wireshark y tshark.

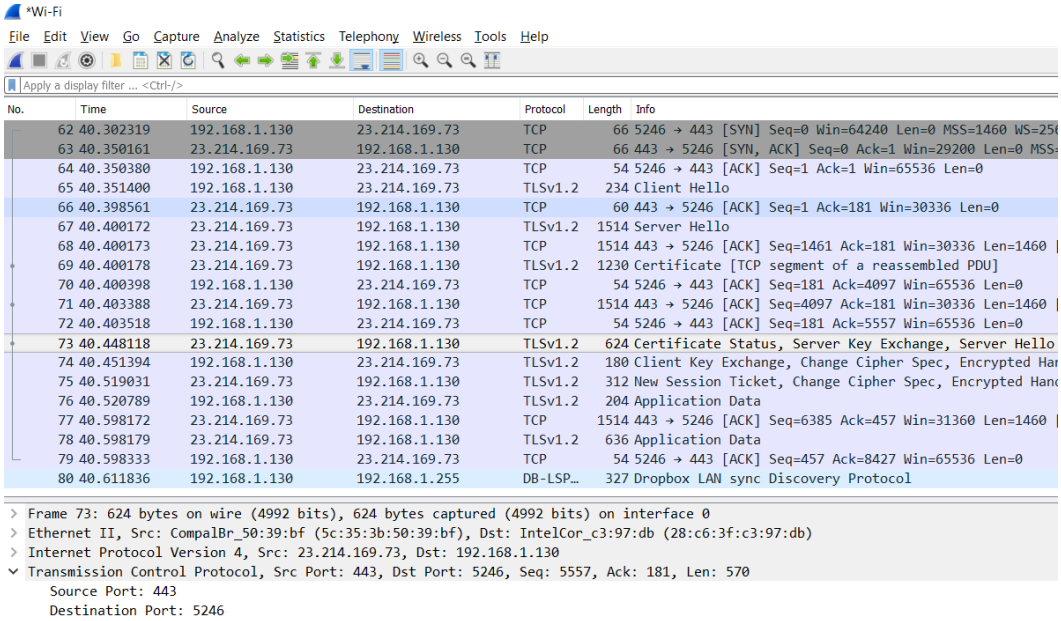


Figura 54: Captura de tráfico de red con Wireshark

https://www.sans.org/reading-room/whitepapers/logging/custom-full-packet-capture-system-34177 http://gauss.ececs.uc.edu/Project4/Documents/nsm.pdf

1.1.2 Datos de sesión

Los datos de sesión proporcionan información mínima y resumida del intercambio de paquetes cuando conexiones son establecidas entre dos dispositivos de la red. Usualmente sesión hace referencia a conexiones establecidas mediante protocolos orientados a la conexión como TCP, pero conexiones bajo entorno solicitud-respuesta también pueden ser consideradas como flujo de datos de sesión.

Existe una mayor facilidad en almacenar los datos de sesión debido que no contienen la información en bruto como ocurre con la captura de paquetes. Se puede fácilmente desplegar una solución el cual almacene estos datos por un periodo mucho mayor que los datos de contenido completo. Por otro lado, los datos de sesión son útiles cuando la red maneja altas tasas de transmisión en donde capturar la totalidad de los paquetes para realizar un análisis podría no ser la mejor opción.

El protocolo mayormente conocido para la recopilación de datos de sesión es Netflow, el cual recopila información tal como:

- Dirección IP origen y destino

- Puerto origen y destino
- Protocolo usado
- Total de datos transferidos
- Duración de la conexión
- Marca de tiempo
- Bandera TCP

Otra herramienta que permite la captura de estos tipos de datos es Zeek (Bro), una plataforma que sera dará mayor detalle en el capitulo XX. La información generada por Zeek es la que se indica a continuación

1.1.3 Datos de estadísticos

Estos datos permiten una rápida identificación y analisis de anomalías o incidentes debido a problema de rendimiento en la red o dispositivos. Por ejemplo, las gráficas de datos estadísticos podrían determinar de una manera sencilla si un dispositivo en una determinada interfaz de red recibe o envía una tasa de trafico anómala. La forma gráfica de estos tipos de datos permiten además resolver problemas de conectividad en una red o ataques orientados a denegación de servicio.

Usualmente las mediciones del trafico en la red se basan en cantidad de paquetes, bytes o bits por segundo. Básicamente los datos estadísticos están relacionados a determinar:

- Tráfico generado en una red
- Longitud de paquetes transmitidos con mayor frecuencia
- Listado de clientes o servicios con mayor cantidad de paquetes enviados

La forma mas común para visualizar las estadísticas del uso de una red es mediante el empleo del protocolo SNMP, el cual permite monitorizar dispositivos de red mediante el intercambio de información sobre el estado de estos.

1.1.4 Datos de alerta

La generación de alertas es una característica de los IDS o de cualquier herramienta de monitorización que envía notificaciones sobre algún evento relevante. Un IDS generará una alerta en el caso de encontrar alguna intrusión o anomalía durante la inspección del tráfico que realiza.

1.1.5 Datos de logs

Los logs es un registro que contiene información relacionada a diferentes tipos de eventos que ocurren en redes o sistemas. Estos datos son útiles para realizar una investigación de actividades maliciosas tales como acceso no autorizado a un sistema, robo de información o uso inapropiado de recursos. Los logs son generados por varias entidades:

- *Sistemas operativos:* generan eventos del sistema tales como errores y estados de servicios. Los sistemas operativos también son capaces de producir logs de auditoría para determinar información relacionada a inicios de sesión fallidos y válidos, modificación de ficheros, cambio de privilegios, etc.
- *Dispositivos de red:* Routers, switches, puntos de accesos, entre otros generan eventos relacionados a su funcionamiento, cambios de configuración, tráfico transferido, etc.
- *Aplicaciones:* las aplicaciones generan diversos tipos de logs basada en su funcionalidad. Los eventos producidos usualmente son relacionados a actividades de usuario, estado, errores y cambios de configuración.
- *Software de seguridad:* dentro de esta categoría se encuentran sistemas de detección y prevención de intrusos, antivirus, firewalls. Estos eventos son considerados de mayor relevancia a los anteriores, ya que son generados al ocurrir algún tipo de ataque, intrusión o infección en la red.

1.2 Detección

La detección se encarga de analizar datos con el fin de identificar patrones anómalos que pueden ser relacionados a un ataque a un sistema o red. El proceso de detección se realiza a través de firmas, anomalías o estadísticas en donde se genera una alerta relacionados a eventos generados.

Los dispositivos empleados para implementar el mecanismo de detección son comúnmente sistemas de detección de intrusos (IDS) y firewalls. Dentro de los IDSs, las soluciones open source más populares son : Snort, Suricata, Bro (Zeek) y OSSEC.

1.2.1 Sistema de detección de intrusos

Un sistema de detección de intrusos monitoriza el tráfico con el objetivo principal de identificar si una entidad ha vulnerado los controles de seguridad de un sistema. Los IDSs generan alertas sobre la detección de un incidente o anomalía, normalmente

basan su detección en la comparación del tráfico capturado con firmas. Las firmas con conjunto de patrones o reglas de empleados para detectar alguna actividad maliciosa en la red.

Existen dos tipos de IDSs según donde sean instalados:

- **IDS basados en host**

  Un sistema de intrusos basados en host (HIDS) analiza actividades maliciosas en hosts específicos en donde haya sido instalado. Un HIDS realiza normalmente análisis de logs del sistema, monitorización de integridad de ficheros y registros. La mayoría de soluciones se implementa a través de la instalación de un agente local, en donde realiza un monitorización a través de reglas y firmas para identificar actividad inusual en el host.

  Una de las soluciones más utilizadas para el despliegue de un HIDS es OSSEC, el cual es una solución *open source*, que además de verificar la integridad de ficheros, auditar y monitorizar registros, permite una detección de rootkits y la exportación de información relevante a SIEMs. La información generada por un HIDS usualmente es de utilidad para realizar una correlación de eventos en un SIEM.

- **IDS basados en red**

  Un sistema de detección de intrusos basado en red (NIDS) tiene una funcionalidad similar al HIDS, en donde la principal diferencia es que la monitorización se realiza por un sensor en un determinado segmento de red. En este caso no se emplea un agente, sino que se requiere de un dispositivo para obtener una copia exacta del tráfico que atraviesa por la red. Estos dispositivos puede ser un TAP, un switch con un puerto SPAN o un hub.

  El NIDS debe ser capaz de monitorizar no solamente el tráfico destinando a su dirección MAC, sino que además, debe escuchar todo el tráfico que cruza por la red. Para esto, es necesario que el NIDS opere en modo promiscuo con el fin de recibir todos los paquetes independientemente del destinatario. Uno de los principales inconvenientes de los NIDSs es que la monitorización se realiza solamente a un segmento de red, por lo que es necesario el despliegue de sensores adicionales para aumentar la cobertura de protección en otros segmentos.

Por otra parte, basado en el mecanismo de detección los IDSs se puede clasificar en:

- **Detección por firmas**

  Es el más antiguo método de detección de los IDSs. Este tipo de detección se basa en la comparación de una base de datos de firmas con el tráfico de la red y generar una alerta en el caso de que exista una coincidencia, de manera similar como ocurre con un antivirus. Una firma es un conjunto de reglas que corresponde a una amenaza conocida, la firma puede basarse en un paquete o en la secuencia de varios. Las reglas usualmente se definen en base a la información relacionada a direcciones IP, cadenas de caracteres, número de bytes, etc.

  Los IDSs basados en firmas pueden detectar ataques conocidos, pero presentan dificultades ante nuevos ataques o variantes en donde no poseen reglas para estos nuevos tipos de instrucciones. Es necesario la actualización periódica de las firmas para la que la detección sea efectiva. Uno de los principales herramientas que se basan en este tipo de detección es Snort.

  Snort

  Snort es un sistema de detección de intrusos basados en red (NIDS) open source, el cual utiliza una combinación de reglas y preprocesadores para analizar el trafico de la red. La arquitectura de Snort se conforma de varios componentes:

  - Descodificador de paquetes (Packet Decoder): tiene la funcione de determinar los protocolos y el tamaño de la carga útil del paquete para que sea usado por el preprocesador el motor de detección. Además, verifica posible errores en las cabeceras y revisa la suma de verificación de cada uno de los paquetes capturados.
  - Preprocesadores (Preprocessors): permiten un análisis más profundo del paquete incluso su modificación. A diferencia de las reglas, los preprocesadores pueden efectuar tareas adicionales tales como: desfragmentación de paquetes, reemsamblaje de secuencias TCP, inspección de sesión, manipulación de flujos TCP, entre otras.
  - Motor de detección (Detection Engine): recibe los paquetes que han sido analizados por los procesadores, estos datos son verificados mediante la base de datos de reglas. En el caso de existir alguna coincidencia con alguna regla del IDS se envían al procesador de alertas.
  - Salida (Output): genera una alerta en el caso de alguna coincidencia detectada

  En la figura 55 se indica los componentes que conforman el IDS Snort.

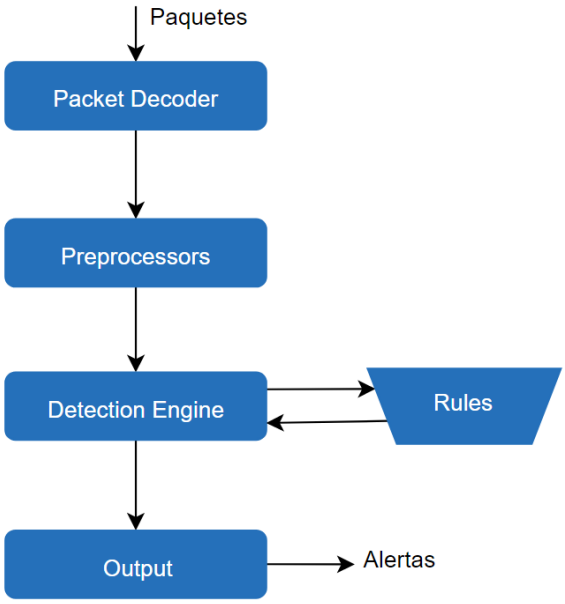


Figura 55: Arquitectura de Snort

- **Detección por anomalías**

  Este tipo de detección realiza un análisis del tráfico para identificar patrones que no coinciden con un conjunto de características o actividades en la red definidas como un comportamiento normal. Actualmente se utilizan técnicas de inteligencia artificial para desarrollar un modelo de aprendizaje para establecer el comportamiento normal, específicamente mediante redes neuronales.

  El mecanismo de detección basado por anomalías es eficaz para detectar nuevos tipos de ataques previamente desconocidas que tienen un comportamiento similar a otros tipos de intrusiones. La complejidad en este tipo de método de detección es definir que es un comportamiento normal en la red. Además, es posible que la detección por anomalías genere un número alto de falsos positivos en comparación al la detección basada en firmas cuando en el patrón de la red se modifique y se produzca una alerta de una anomalía.

  Zeek es una herramientas que basa su funcionamiento en detección de anomalías y el cual se basa este proyecto para la generación de alertas de seguridad.

  Zeek (Bro)

  Zeek es una herramienta open source de monitorización de seguridad de red, el cual inspecciona el tráfico de red en tiempo real. En esta herramienta la

totalidad de las conexiones, sesiones e incluso información a nivel de capa de aplicación son enviadas a logs para su posterior análisis. Los logs generados contienen no sólo un registro completo de cada conexión, sino también, todas las sesiones HTTP con sus URIs solicitadas, encabezado, solicitudes DNS con respuestas, certificados SSL, entre otros.

Aunque, este software también puede trabajar con firmas importadas de Snort, el funcionamiento de Zeek es diferente a un IDs basado en firmas. Zeek utiliza un lenguaje de scrpting para definir un conjunto de políticas de seguridad e indicar que acción tomar al detectar alguna actividad anómala en la red. El principal beneficio del lenguaje tipo script basado en eventos es permitir la ampliación y personalización de las funcionalidades de la solución. Aunque cuenta con un conjunto de funcionalidades previamente cargadas, es posible definir políticas especificas que se adapte a un entorno particular de red según las necesidades.

Su funcionamiento se basa en dos componentes principales:

- Motor de eventos (event engine): analiza el tráfico capturado para generar eventos de nivel superior. Los eventos generados son neutrales a la políticas definidas, solamente señalen algún tipo de actividad que ocurrió, es decir describe que algo sucedió en la red y so establece una interpretación del evento. Por ejemplo no se establece si un dominio o URL corresponde a un sitio malicioso.
- Intérprete script (script interpreter): ejecuta una serie de manejadores de eventos definidos en los scripts. Se establece que acciones tomar cuando se detecta alguna actividad. Zeek puede generar logs, ejecutar programas externos, enviar una notificación, etc.

Zeek al realizar un análisis con mayor profundidad de los paquetes, identifica una gran variedad de protocolos incluso los que no se ejecutan en los puertos estándar a través de la función denominada Detección de Protocolo Dinámico (DPD).

En la figura 56 se puede visualizar la arquitectura de Zeek en donde constan los dos principales componentes descritos.

Las principales funcionalidades y características de Zeek se pueden resumir:

- Realiza análisis constante y seguimiento del tráfico HTTP.

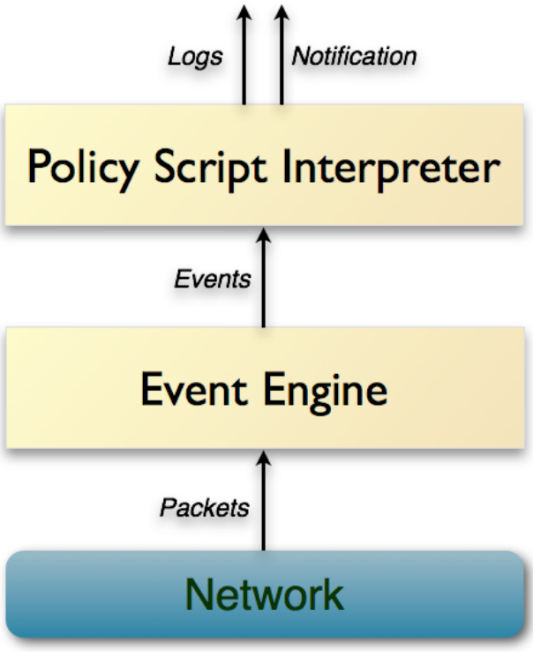


Figura 56: Arquitectura de Zeek [12]

- Es capaz de detectar ataques de fuerza bruta contra servicios de red tales como SSH y FTP
- Ejecuta validaciones en los certificados SSL / TLS
- Analiza y detecta cambios en el software instalado en la máquina
- Genera reportes usando email
- Detección y análisis de túneles (incluyendo Ayiya, Teredo, GTPv1)
- Bro desencapsula túneles y luego procede a analizar su contenido como si no hubiera ningún túnel en el lugar.
- Detecta ataques de SQL

1.3 Análisis

Esta etapa esta relacionada al proceso de interpretación y validación de alertas o eventos maliciosos para determinar si un incidente ha ocurrido. En analista se encarga de recopilar toda la información necesaria de varias fuentes para determinar el alcance del incidente de seguridad y si realmente existe el dicho incidente, es decir que no se trate de un falso positivo. Las fuentes externas también son útiles para realizar el análisis, por ejemplo un reporte o aviso de un CERT u otra entidad.

Normalmente las alertas generados por IDSs son investigadas por el analista. En este caso, la primera tarea una vez descartada la opción de falso positivo es recopilar información acerca de las entidades involucradas, tales como dirección IP origen y destino y naturaleza del incidente detectado.

El análisis puede incluir las siguientes tareas:

- Análisis de paquetes
- Análisis de datos de inteligencia
- Análisis forense
- Análisis de malware

Estas tareas permiten identificar las comunicaciones establecidas entre involucrados, establecer puertos y protocolos. El principal objetivo en esta etapa es determinar como ha ocurrido las actividades maliciosas y toda la conexiones establecidas por los involucrados o afectados.

Por otro lado, uno de los aspectos importantes en el análisis es realizar una clasificación de los incidentes. La categorización de los incidentes son realizados de acuerdo a nivel de criticidad, impacto, violaciones normativas o legales, etc. Esto permite una mejor gestión de incidentes cuando exista varios eventos que se producen al mismo tiempo con el fin de asignar prioridades a estos incidentes para su escalado y resolución .

2 Docker

Docker es una plataforma que posee la capacidad de empaquetar aplicaciones en conjunto con sus dependencias y ejecutar en un entorno aislado denominado contenedor. Es decir, los contenedores se ejecutan directamente en el kernel del equipo anfitrión, es otras palabras proporciona una virtualización del sistema operativo. Los contenedores comparten entre si el kernel del sistema y se ejecutan como un proceso aislado en un espacio de usuario en el equipo anfitrión.

Se cree de manera errónea que el uso de contenedores proporciona funcionalidades idénticas a la virtualización de hardware, pero esto no es así. Docker aísla los

procesos de las aplicaciones y todos los contenedores creados se ejecutan en el mismo sistema host, es decir los contenedores se ejecutan sobre el mismo kernel. La ejecución de varias aplicaciones sobre un sistema anfitrión no representa un aumento de sobrecarga en comparación sobre un entorno de maquinas virtuales que se ejecutan sobre varios sistemas operativos.

Entre las principales ventajas que ofrece el despliegue de contenedores mediante Docker están:

- Ejecución multiplataforma: servicios y aplicaciones se pueden empaquetar en una imagen portátil para la ejecución en múltiples plataformas

- Aislamiento: Los contenedores aíslan el entorno de ejecución entre las aplicaciones. Además se proporciona una capa adicional de seguridad a las aplicaciones

2.1 Arquitectura

Docker se basa en una arquitectura cliente-servidor. El demonio Docker actúa como servidor y es el responsable de crear contenedores y gestionar el ciclo de vida. La comunicación entre el cliente y servidor docker se realiza a través de una API REST, sockets UNIX o una interfaz de red.

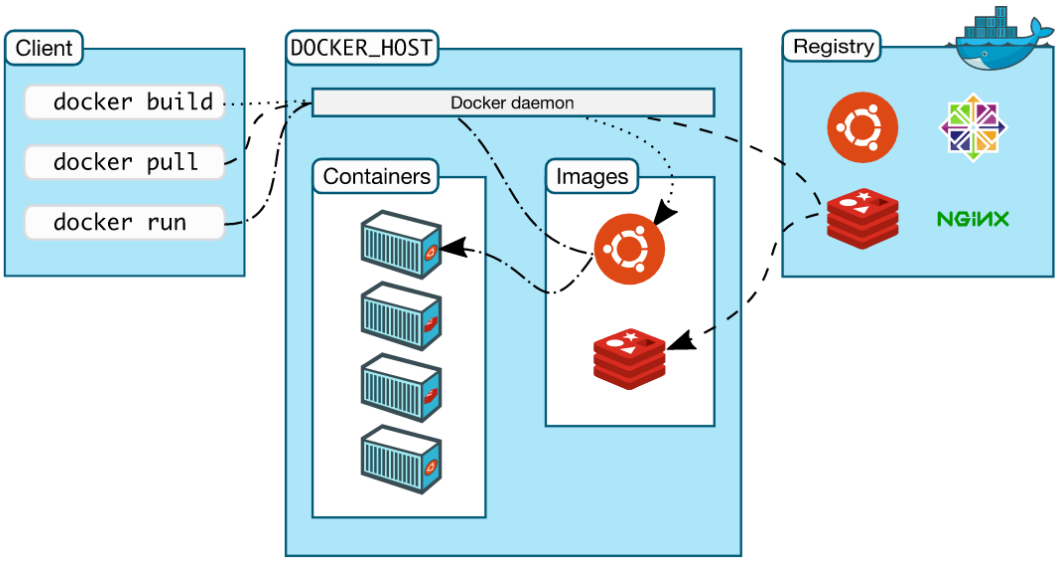


Figura 57: Arquitectura de Docker [25]

- **Cliente Docker:** es la interfaz entre el el usuario y el demonio Docker a través de una serie de comandos.

  Los principales comandos utilizados para la administración de contenedores son:

Comando	Descripción
docker build	Crear una imagen desde un Dockerfile
docker run	Ejecutar una imagen Docker como un contenedor
docker commit	Crear una nueva imagen a partir de los cambios en un contenedor
ocker pull	Extraer una imagen o un repositorio de un registro

Tabla 9: Comandos en CLI Docker

- **Demonio Docker:** es el componente principal de Docker, es el encargado de gestionar los contenedores (borrar, crear, iniciar o mover), volúmenes, imágenes y redes.
- **Registros Docker:** es un repositorio de imágenes. Docker Hub es el repositorio público en donde se puede encontrar varias imágenes para diversas aplicaciones o servicios
- **Objetos Docker**
  - **Imágenes:** son plantillas de lectura que se conforman a partir de un conjunto de instrucciones escritas en el Dockerfile para la creación de contenedores. La imagen define los procesos que la aplicación empaquetada debe ejecutar así como sus dependencias
  - **Contenedores:** es una instancia ejecutable de una imagen. Los contenedores en Docker poseen su propio sistema de archivos en capas dentro de la máquina host. Varios contenedores pueden ser creados desde una imagen dentro de un entorno aislado..

Al momento de eliminar un contenedor, también la capa de escritura es removida, por lo que lo archivos y modificaciones creados dentro del contenedor no son accesibles y desaparecen. Docker posee varias opciones para que los datos persistan una vez que el contenedor sea removido mediante el uso de:

- Volúmenes
- Sistema de almacenamiento tmpfs
- Bind mounts

2.2 Volúmenes

Los volúmenes es el método más recomendado para conservar los datos dentro de Docker, cuando el contenedor se detiene o remueve el volumen sigue existiendo. Los volúmenes son almacenados en un parte del sistema de ficheros del equipo anfitrión. Al momento de crear un volumen estos son administrados por Docker y aislados de la máquina host.

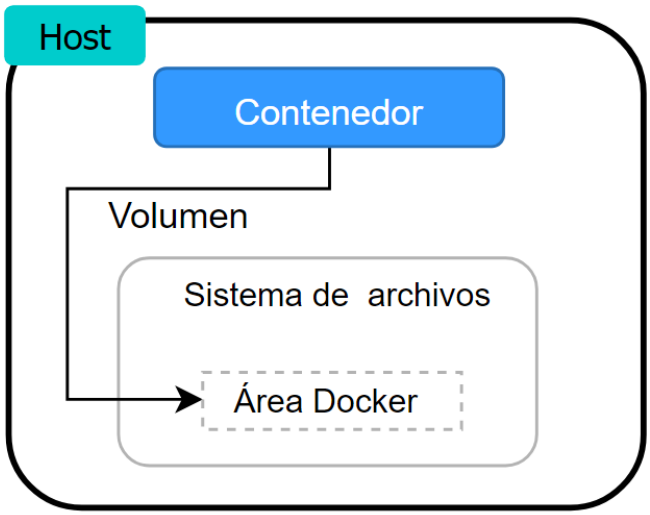


Figura 58: Volumen en Docker

La principal ventaja de manejo de volúmenes es que pueden ser compartidos entre múltiples contenedores y ejecutados simultáneamente. Por lo tanto, es posible compartir datos entre contenedores en funcionamiento. Además, el manejo de volúmenes facilita realizar respaldos, migrar o restaurar de datos hacia otro sistema Docker.

La creación de un volumen se la puede realizar mediante instrucciones en el fichero Dockerfile y a través de linea de comando al iniciar un contenedor. Por ejemplo el siguiente fragmento del fichero Dockerfile permite la creación de un punto de montaje en */data* y copiar el archivo de *test* en el volumen creado.

```
FROM docker.io/centos:6
RUN mkdir /data
RUN echo "hola mundo" > /data/test
VOLUME /data
```

SISTEMA DE GESTIÓN DE EVENTOS RELACIONADOS CON ANOMALÍAS EN LA SEGURIDAD EN UN ENTORNO OPEN SOURCE BASADOS EN DOCKER

ANEXOS

Suministrador:
Alberto Mejía Viteri
alberto.mejiaviteri@alum.uca.es

Puerto Real, septiembre 2019

Anexos

1 Anexo 1: Instalación de herramientas

1.1 Zeek

Zeek requiere de las siguientes librerías y herramientas para su funcionamiento

- Libpcap
- OpenSSL
- BIND8
- Libz
- Python 2.6 o mayor
- Bash

1. Actualizar el listado de paquetes disponibles

```
sudo apt-get update
```

2. Instalar paquetes y librerías necesarias

```
$ sudo apt-get install bison cmake flex g++ gdb make libmagic-dev
     libpcap-dev libgeoip-dev libssl-dev python-dev swig2.0 zlib1g
    -dev
```

3. Instalar Zeek desde código fuente obtenido del repositorio de GitHub

```
$ git clone --recursive https://github.com/zeek/zeek
$ cd bro
$ ./configure
$ make
$ sudo make install
```

Zeek será instalado en el directorio por defecto: /usr/local/bro

4. Configurar la variable de entorno

```
$ export PATH=$PATH:/usr/local/bro/bin
```

Los ficheros de configuración están ubicados en el directorio /usr/local/bro/etc:

- node.cfg
- networks.cfg
- broctl.cfg

El inicio del sistema se lo puede realizar utilizando el shell interactivo denominado *BroControl* mediante el comando:

```
broctl
```

La instancia Zeek es iniciada así:

```
[BroControl] > start
```

1.2 Agentes

La instalación de Auditbeat, Metricbeat y Filbeat son similares, por lo que a continuación se explica la instalación de solamente uno de estos, Filebeat.

1. Importar repostorio mediante llave publica RGP

```
wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch |
    sudo apt-key add -
```

2. Crear el repositorio, en donde se añade un lista de fuente a sources.list

```
$ echo "deb https://artifacts.elastic.co/packages/7.x/apt stable
    main" | sudo tee -a /etc/apt/sources.list.d/elastic-7.x.list
```

3. Actualizar la lista de paquetes

```
$ sudo apt update
```

4. Instalar Filebeat

```
$ apt-get install apt-transport-https
$ apt install filebeat
```

5. Verificar la instalación

```
$ apt-cache policy filebeat
```

```
filebeat:
 Installed: 7.1.1
 Candidate: 7.2.0
 Version table:
    7.2.0 500
       500 https://artifacts.elastic.co/packages/7.x/apt stable/main
           amd64 Packages
*** 7.1.1 500
       500 https://artifacts.elastic.co/packages/7.x/apt stable/main
           amd64 Packages
```

1.3 Instalación de Docker

La instalación de Docker se realizó sobre un máquina Ubuntu server 18.04.2

1. Actualizar el listado de paquetes disponibles

```
sudo apt-get update
```

2. Instalar paquetes necesarios

```
$ sudo apt install apt-transport-https ca-certificates curl
    software-properties-common
```

3. Agregar la clave GPG de Docker

```
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo
    apt-key add -
```

4. Añadir el repositorio de Docker

```
$ sudo add-apt-repository "deb [arch=amd64] https://download.
    docker.com/linux/ubuntu bionic stable"
```

5. Actualizar el repositorio

```
$ sudo apt-get update
```

6. Instalar Docker

```
$ sudo apt install docker-ce
```

1.3.1 Docker Compose

Docker Compose es una herramienta que permite el despliegue de múltiples contenedores y el uso de archivos YAML para la configuración de servicios.

1. Descargar la última versión estable de Docker compose

```
$ sudo curl -L "https://github.com/docker/compose/releases/
    download/1.24.0/docker-compose-$(uname -s)-$(uname -m)" -o /
    usr/local/bin/docker-compose
```

2. Otorgar los permisos necesarios al binario descargado

```
$ sudo chmod +x /usr/local/bin/docker-compose
```

3. Verificar la instalación

```
 $ docker-compose --version
```

1.3.2 Imagen

1. Clonar repositorio de github

```
 $ git clone https://github.com/deviantony/docker-elk.git
```

2. Arrancar contenedores una vez que los cambios en las configuración sean realizados

```
 $ docker-compose up -d
```

1.4 Feed datos de inteligencia

1. Instalar Python

```
$ sudo wget https://www.python.org/ftp/python/2.7.16/Python
   -2.7.16.tgz
$ sudo tar xzf Python-2.7.16.tgz
$ cd Python-2.7.16
$ sudo ./configure --enable-optimizations
$ sudo make altinstall
```

2. Instalar pip

```
$ sudo apt install python-pip
```

3. Instalar el Gadget csirtg smrt para el consumo de los datos de inteligencia

```
$ sudo pip install csirtg-smrt
```

4. Obtener una copia de fichero de configuración

```
$ curl https://raw.githubusercontent.com/csirtgadgets/csirtg-
   smrt-py/master/examples/csirtg.yml > csirtg.yml
```

5. Ejecutar csirtg-smrt

```
$ csirtg-smrt -r csirtg.yml -f FEED --format zeek
```

2 Anexo 2: Logs Zeek

2.1 intel.log

En el momento que Zeek detecte una URL o dominio que esta presente en uno de los *feeds* el *Framework* de Inteligencia generará información en el fichero intel.log tal como se indica en la siguiente salida.

```
#fields ts uid id.orig_h id.orig_p id.resp_h id.resp_p seen.indicator ↩
    ↪ seen.indicator_type seen.where seen.node matched sources fuid ↩
    ↪ file_mime_type file_desc
#types time string addr port addr port string enum enum string set[enum↩
    ↪ ] set[string] string string string
1563489326.741063 CgHJoC3lOIfNGvVSTj 192.168.2.20 35058 8.8.8.853 ↩
    ↪ adtesserver.com Intel::DOMAIN DNS::IN_REQUEST bro Intel::DOMAIN ↩
    ↪ CIF - need-to-know - - -
1563489326.749018 CZUOvv1oK8p4aX7ey1 192.168.2.20 38068 8.8.8.853 ↩
    ↪ adtesserver.com Intel::DOMAIN DNS::IN_REQUEST bro Intel::DOMAIN ↩
    ↪ CIF - need-to-know - - -
1563489326.792658 CgHJoC3lOIfNGvVSTj 192.168.2.20 35058 8.8.8.853 ↩
    ↪ adtesserver.com Intel::DOMAIN DNS::IN_REQUEST bro Intel::DOMAIN ↩
    ↪ CIF - need-to-know - - -
1563489326.809290 CZUOvv1oK8p4aX7ey1 192.168.2.20 38068 8.8.8.853 ↩
    ↪ adtesserver.com Intel::DOMAIN DNS::IN_REQUEST bro Intel::DOMAIN ↩
    ↪ CIF - need-to-know - - -
```

Los campos de mayor relevancia que se identifican dentro del contenido de fichero `intel.log` se indican en la tabla 10.

2.2 notice.log

Detecta anomalías en le red como por ejemplo intentos de ataques, escaneo de red, entre otros. La siguiente salida corresponde a una anomalía relacionada con una exploración de vulnerabilidades inyección SQL a un activo determinados, el registro permite la identificación tanto al atacante como a la víctima.

```
#fields ts uid id.orig_h id.orig_p id.resp_h id.resp_p fuid ↩
    ↪ file_mime_type file_desc proto note msg sub src dst …
#types time string addr port addr port string string string enum enum ↩
    ↪ string string addr addr port …
1560520839.807443 - - - - - - - - - HTTP::SQL_Injection_Attacker An SQL↩
    ↪  injection attacker was discovered! - 192.168.2.4 - - - - Notice↩
```

Item	Campo	Tipo	Descripción
1	ts	tiempo	Timestamp en formato UNIX
2	uid	string	Identificador único de conexión
3	id.orig_h	addr	Dirección IP origen
4	id.orig_p	port	Número de puerto origen
5	id.resp_h	addr	Dirección IP destino
6	id.resp_p	port	Número de puerto destino
7	seen.indicator	string	Dominio, URL, IP, etc
8	seen.indicator_type	enum	El tipo del indicador
9	seen.where	enum	Donde fueron descubiertos los datos
10	seen.node	string	El nodo donde se descubrió la coincidencia
11	matched	set[enum]	Los tipos de indicador que coinciden.
12	sources	set[string]	Fuentes que proporcionaron los datos

Tabla 10: Campos del registro intel.log

```
    ↪ ::ACTION_...
1560520839.807443 - - - - - - - - - HTTP::SQL_Injection_Victim An SQL ↩
    ↪ injection victim was discovered! - 192.168.2.8 - - - - Notice::↩
    ↪ ACTION_LOG …
```

Los campos de mayor relevancia que se identifican dentro del contenido de fichero `notice.log` se indican en la tabla 11.

2.3 conn.log

A continuación, se presenta un ejemplo de la información almacenada en el fichero `conn.log`.

```
#fields ts uid id.orig_h id.orig_p id.resp_h id.resp_p proto service ↩
    ↪ ...
#types time string addr port addr port enum string interval count count↩
    ↪  ..
1560212579.634621 Cictqm3ihctjSHYSU2 192.168.2.50 45798 91.189.92.38 ↩
    ↪ 443 tcp ...
1560212576.046813 C72yLcma3HShGPFMg 192.168.2.50 51873 8.8.8.8 53 udp ↩
    ↪ dns ...
```

Item	Campo	Tipo	Descripción
1	ts	tiempo	Timestamp en formato UNIX
2	uid	string	Identificador único de conexión
3	note	enum	Notificación
4	msg	string	Mensaje en formato legible
5	src	addr	Dirección IP fuente
6	dst	addr	Dirección IP destino
7	p	port	Puerto
8	n	count	Código de estado

Tabla 11: Campos del registro notice.log

```
1560212576.046810 Cq911z1LC8UsvdJQv5 192.168.2.50 35338 8.8.8.8 53 udp ↩
    ↪ dns ...
1560212578.687589 CehXEagnPOH2nv033 192.168.2.50 54590 8.8.8.8 53 udp ↩
    ↪ dns ...
```

Los campos de mayor relevancia que se identifican dentro del contenido de fichero `conn.log` se indican en la tabla 12.

Item	Campo	Tipo	Descripción
1	ts	tiempo	Timestamp en formato UNIX
2	uid	string	Identificador único de conexión
3	id.orig_h	addr	Dirección IP origen
4	id.orig_p	port	Número de puerto origen
5	id.resp_h	addr	Dirección IP destino
6	id.resp_p	port	Número de puerto destino
7	proto	enum	Protocolo
8	service	string	Servicio

Tabla 12: Campos de conn.log

SISTEMA DE GESTIÓN DE EVENTOS RELACIONADOS CON ANOMALÍAS EN LA SEGURIDAD EN UN ENTORNO OPEN SOURCE

ESPECIFICACIONES DEL SISTEMA

Suministrador:
Alberto Mejía Viteri
alberto.mejiaviteri@alum.uca.es

Puerto Real, septiembre 2019

Especificaciones del sistema

1 Objetivos

Los objetivos que se han definido en el proyecto son los que se indican el la tabla 13:

Objetivo	Descripción
OB-01	Establecer un sistema de detección de intrusos para la generación de anomalías
OB-02	Definir diferentes tipos de fuentes de información para su análisis
OB-03	Implementar un sistema de envío, recopilación de eventos y registros
OB-04	Desplegar un sistema de almacenamiento de la información procesada
OB-05	Instalar un sistema de visualización y análisis de eventos de seguridad
OB-06	Monitorizar el sistema de gestión de eventos

Tabla 13: Objetivos definidos en el proyecto

2 Especificaciones

El sistema desplegado debe cumplir con las siguientes especificaciones:

- R-01: Generar anomalías de la red mediante el análisis del tráfico
- R-02: Recopilar datos de contenido completo y de sesión
- R-03: Recopilar datos sesión

- R-04: Recopilar datos de estadísticos auditoría
- R-05: Recopilar datos de auditoría
- R-06: Enviar eventos desde las fuentes generadoras
- R-07: Normalizar eventos para su almacenamiento
- R-08: Almacenar la información generada
- R-09: Visualizar información para análisis
- R-10: Cifrar datos entre emisor y receptor de eventos
- R-11: Monitorizar el sistema de gestión de eventos

R-01	Generar anomalías de la red mediante el análisis de tráfico
Objetivos asociados	OB-01
Requisitos asociados	NA
Descripción	Mecanismo que permite la inspección de paquetes en tiempo real para la generación de anomalías

Tabla 14: Requisito 1 del sistema

R-02	Recopilar datos de contenido completo
Objetivos asociados	OB-02
Requisitos asociados	NA
Descripción	Captura de tráfico para generar datos de contenido completo

Tabla 15: Requisito 2 del sistema

R-03	Recopilar datos de sesión
Objetivos asociados	OB-02
Requisitos asociados	NA
Descripción	Captura de tráfico para generar datos de sesión

Tabla 16: Requisito 3 del sistema

R-04	Recopilar datos estadísticos
Objetivos asociados	OB-02
Requisitos asociados	NA
Descripción	Monitorización local mediante sensores o agentes

Tabla 17: Requisito 4 del sistema

R-05	Recopilar datos de auditoría
Objetivos asociados	OB-02
Requisitos asociados	NA
Descripción	Monitorización local mediante sensores o agentes

Tabla 18: Requisito 5 del sistema

R-06	Enviar de eventos desde las fuentes generadoras
Objetivos asociados	OB-03
Requisitos asociados	R-02, R-03, R-04 y R-05
Descripción	Definir un mecanismo de envió de información relevante de seguridad

Tabla 19: Requisito 6 del sistema

R-07	Normalizar eventos para su almacenamiento
Objetivos asociados	OB-03
Requisitos asociados	R-06
Descripción	Aplicar filtros que permitan la transformación a un formato común de eventos necesarios

Tabla 20: Requisito 7 del sistema

R-08	Almacenar la información generada
Objetivos asociados	OB-04
Requisitos asociados	NA
Descripción	Almacenamiento de información para el consumo posterior

Tabla 21: Requisito 8 del sistema

R-09	Visualizar información para análisis
Objetivos asociados	OB-05
Requisitos asociados	NA
Descripción	Mecanismo de visualización para análisis a través de gráficas y métricas

Tabla 22: Requisito 9 del sistema

R-10	Cifrar datos entre emisor y receptor de eventos
Objetivos asociados	OB-03
Requisitos asociados	R6
Descripción	Cifrado de la comunicación de las entidades de envío y recopilación de eventos

Tabla 23: Requisito 10 del sistema

R-11	Monitorizar el sistema de gestión de eventos
Objetivos asociados	OB-06
Requisitos asociados	NA
Descripción	Monitorización de recursos del sistema desplegado

Tabla 24: Requisito 11 del sistema

3 Matriz de rastreabilidad

En la siguiente tabla se establece la matriz de rastreabilidad del presente proyecto

	OB-01	**OB-02**	**OB-03**	**OB-04**	**OB-05**	**OB-06**
R-01	•					
R-02		•				
R-03		•				
R-04		•				
R-05		•				
R-06		•	•			
R-07		•	•			
R-08				•		
R-09					•	
R-10			•			
R-011						•

Tabla 25: Matriz de rastreabilidad

SISTEMA DE GESTIÓN DE EVENTOS RELACIONADOS CON ANOMALÍAS EN LA SEGURIDAD EN UN ENTORNO OPEN SOURCE BASADOS EN DOCKER

MEDICIONES

Suministrador:
Alberto Mejía Viteri
alberto.mejiaviteri@alum.uca.es

Puerto Real, septiembre 2019

Mediciones

En este apartado se realizan mediciones necesarias para definir el presupuesto del proyecto. Las mediciones se efectúan tomando en cuenta el despliegue del proyecto en un entono de producción.

Es importante definir los recursos adecuados que posee el servidor que actuará como analizador de trafico, en este caso a través de la plataforma Zeek. Tal como se mencionó anteriormente, se establece un cantidad de 250 Mbps de trafico analizado por cada núcleo de CPU. En el caso de se pretenda desplegar esta solución sobre una red que alcanza un pico máximo de trafico de 1 Gbps se requiere una cantidad de al menos 4 núcleos en el servidor. Con respecto a cantidad de RAM se recomienda lo máximo como sea posible. Por otro lado el servidor dedicado para la implementación de Elastic Stack debe tener características similares al servidor Zeek igualmente debido a la cantidad de procesamiento que realiza al analizar información de diferentes fuentes.

Además, un elemento de importante en una red donde se efectuar un análisis del trafico es el TAP, con el fin de obtener un replica del trafico de la red analizada. La velocidad máxima de transmisión de este dispositivo debe ser igual a la interfaz de red del servidor Zeek.

Ítem	Elemento	Unidades
1	Servidor DELL PowerEdge R440	2
2	TAP Datacom CTP-1000	1
3	Material cableado	1

Tabla 26: Medición de equipamiento/material

Ítem	Elemento	Horas
1	Mano de obra	160

Tabla 27: Medición de mano de obra

SISTEMA DE GESTIÓN DE EVENTOS RELACIONADOS CON ANOMALÍAS EN LA SEGURIDAD EN UN ENTORNO OPEN SOURCE BASADOS EN DOCKER

PRESUPUESTO

Suministrador:
Alberto Mejía Viteri
alberto.mejiaviteri@alum.uca.es

Puerto Real, septiembre 2019

Presupuesto

1 Equipamiento

Ítem	Descripción	Cantidad	Coste unitario €	Coste Total €
1	**Servidor Zeek & Elastic Stack** DELL PowerEdge R440 - Intel Xeon Silver - RAM: 16 GB - Núcleos: 8 - Disco: 600GB - NIC: Dual 1GbE	2	1675.00	3350.00
2	**Network TAP** Datacom CTP-1000 - Auto negociación - Interfaces PoE - Paquetes hasta 10240 B	1	990.00	990.00
3	Cableado	1	50.00	50.00
		Total	**4390.00**	

Tabla 28: Presupuesto de equipamiento/material

2 Mano de obra

Item	Descripción	Cantidad horas	Coste unitario €	Coste Total €
1	Mano de obra - Instalación - Implementación - Pruebas de funcionamiento	160	15	2400
		Total	**2400**	

Tabla 29: Presupuesto mano de obra

3 Resumen

Ítem	Descripción	Coste Total (€)
1	Equipamiento	4390.00
2	Mano de obra	2400.00
	Total	**6790.00**

Tabla 30: Presupuesto del proyecto